Открытая олимпиада школьников (информатика)
(№62 Перечня олимпиад школьников, 2022/2023 уч.год)

Содержание

Задания для 11 класса

Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (3 балла)

[A и B сидели на трубе]

Определите две последние цифры шестнадцатеричной записи числа $0,AB_{16}^N$ при N=20 000 000 000$_{10}$.

В ответе запишите подряд две шестнадцатеричные цифры в порядке их следования в числе.

Ответ: 01

Решение:

Докажем, что $0,AB_{16}^N = \frac{AB_{16}^N}{100_{16}^N}$, например так:

$0,AB_{16}^N = X$

$0,AB_{16}^N * 100_{16}^N = X * 100_{16}^N$

$(0,AB * 100)_{16}^N = X * 100_{16}^N$

$AB_{16}^N = X * 100_{16}^N$

$X = \frac{AB_{16}^N}{100_{16}^N}$

$0,AB_{16}^N = \frac{AB_{16}^N}{100_{16}^N}$

Обратим внимание, что деление на 100_{16}^N приведет просто к изменению позиции запятой и, следовательно, последние две цифры шестнадцатеричной записи числа $0,AB_{16}^N$ будут ровно такими же, как последние две цифры числа AB_{16}^N.

Определение значения AB_{16}^N при N=20 000 000 000 «в лоб» вычислительно затруднено. Но легко доказать, что если построить ряд вида AB_{16}^1, AB_{16}^2, AB_{16}^3, …,

то последние две цифры шестнадцатеричных записей этих чисел будут образовывать период. Действительно, значения последних двух цифр очередного члена ряда будут зависеть только от значений двух последних цифр предыдущего члена ряда, а поскольку количество комбинаций этих цифр конечно, рано или поздно возникнет пара цифр, которые уже ранее встречались, и начнется новый период.

Получить такой период для AB_{16}^N можно, например, программно. Он будет включать в себя 64 значения:

AB, 39, 13, B1, 3B, 69, 23, 61, CB, 99, 33, 11, 5B, C9, 43, C1, EB, F9, 53, 71, 7B, 29, 63, 21, 0B, 59, 73, D1, 9B, 89, 83, 81, 2B, B9, 93, 31, BB, E9, A3, E1, 4B, 19, B3, 91, DB, 49, C3, 41, 6B, 79, D3, F1, FB, A9, E3, A1, 8B, D9, F3, 51, 1B, 09, 03, 01.

Заметим, что следующим значением снова будет AB, что начнет новый период.

Тогда для решения задачи нам нужно вычислить остаток от деления N на 64 (при N=20 000 000 он равен 0) и взять из получившегося набора значений значение с соответствующим номером. Поскольку 20 000 000 делится на 64 нацело, необходимо взять последнее значение. Таким образом, получается ответ «01».

2. Кодирование информации. Объем информации (2 балла)

[4x4]

На занятиях кружка по информатике Михаила Валерьевича Петя и Вася получили задание написать генератор квадратных растровых изображений размером N x N пикселей, где N всегда кратно четырем. Каждый пиксель может быть или черным или белым. Петя решил сохранять в память изображение как последовательность кодов цветов пикселей, используя для записи кода цвета каждого пикселя минимально возможное, одинаковое для всех кодов цветов пикселей. количество бит. Вася заметил особенность генератора Пети – любое получившееся изображение можно разбить на непересекающиеся квадраты размером 4 x 4 пикселя и в каждом таком квадрате всегда получается одинаковое количество белых и черных пикселей. Тогда Вася предложил присвоить уникальный числовой код каждому удовлетворяющему этому условию квадрату 4 x 4 и сохранять в память изображение как последовательность таких кодов, используя для записи кода каждого квадрата минимально возможное, одинаковое для всех кодов квадратов количество бит.

Вася обнаружил, что при его способе записи изображение занимает в памяти на 81 байт меньше, чем при способе записи Пети. Определите N, при котором это возможно. В ответе укажите целое число.

Ответ:72

Решение:

Поскольку каждый пиксель может быть только одного из двух цветов, на хранение его кода потребуется 1 бит. Тогда, при способе Пети изображение будет занимать N^2 бит памяти.

Если изображение разбить на квадраты 4 x 4 пикселя, то всего изображение будет состоять из $\left(\frac{N}{4}\right)^2$ таких квадратов. Поскольку каждый квадрат состоит из 16 пикселей и должен содержать равное количество черных и белых пикселей (то есть по 8), для определения количества вариантов таких квадратов можно воспользоваться формулой для подсчета количества перестановок с повторениями: $\frac{16!}{8!\cdot 8!}$. Теперь можно составить и решить уравнение, описывающее условие задачи, не забыв перевести разницу в объемах занимаемой памяти из байт в биты.

$$N^2 - \left(\frac{N}{4}\right)^2 \cdot \left\lceil log_2\left(\frac{16!}{8! \cdot 8!}\right)\right\rceil = 81 \cdot 8$$

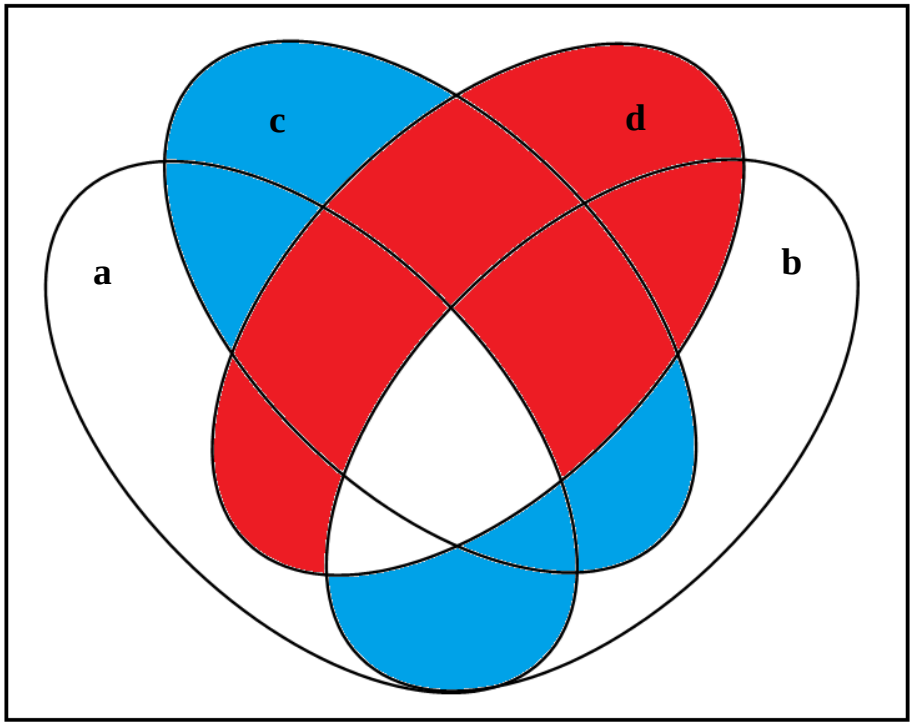
$$N^2 - \left(\frac{N}{4}\right)^2 \cdot \lceil log_2(12870)\rceil = 648$$

$$N^2 - \left(\frac{N}{4}\right)^2 \cdot 14 = 648$$

$$\frac{1}{8}N^2 = 648$$
$$N^2 = 5184$$
$$N = 72$$

3. Основы логики (2 балла)
[Красное и синее]

Дана диаграмма Венна для четырех множеств: a, b, c и d.



Определим логические выражения A, B, C, D как утверждения, что точка принадлежит множеству a, b, c или d, соответственно.

Найдите логическую функцию от четырех аргументов F(A, B, C, D) такую, что она эквивалентна утверждению «Если точка принадлежит красной области, то она принадлежит синей области». В ответе укажите логическую формулу в максимально упрощенном виде, которая может содержать логические переменные A, B, C, D и логические операции из набора {инверсия, конъюнкция, дизъюнкция}. Если таких функций нет, запишите в ответ NULL.

Комментарий по вводу ответа: операнды вводятся большими латинскими буквами; логические операции обозначаются, соответственно, как not, and и or. Запись не должна содержать скобок. Пример записи ответа: A or not B

Ответ: not D or A and B || not D or B and A || A and B or not D || B and A or not D

Решение:

Построим таблицу истинности. Выделим цветом строки, соответствующие окрашенным областям на диаграмме Венна:

A	B	C	D	красная	синяя
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	0	0
1	1	1	0	0	1
1	1	1	1	0	0

Добавим столбец, в котором укажем истинность выражения «точка принадлежит красной области» → «точка принадлежит синей области», то есть значения функции F(A, B, C, D). Обратим внимание, что указанное утверждение является импликацией X → Y, и, следовательно, имеет таблицу истинности:

X	Y	X → Y
0	0	1
0	1	1
1	0	0
1	1	1

Тогда получится следующая таблица:

A	B	C	D	красная	синяя	F(A, B, C, D)
0	0	0	0	0	0	1
0	0	0	1	1	0	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	1	0	0
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	0	0	0	1	1
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	0	0	1

Составим СКНФ для искомой функции:

$$(A \vee B \vee C \vee \bar{D}) \wedge (A \vee B \vee \bar{C} \vee \bar{D}) \wedge (A \vee \bar{B} \vee C \vee \bar{D}) \wedge (A \vee \bar{B} \vee \bar{C} \vee \bar{D}) \wedge (\bar{A} \vee B \vee C \vee \bar{D}) \wedge (\bar{A} \vee B \vee \bar{C} \vee \bar{D})$$

Вынесем $\bar{D}$ за скобки, поскольку оно входит во все выражения:

$$\bar{D} \vee \big((A \vee B \vee C) \wedge (A \vee B \vee \bar{C}) \wedge (A \vee \bar{B} \vee C) \wedge (A \vee \bar{B} \vee \bar{C}) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee B \vee \bar{C})\big)$$

Применим закон склеивания попарно к первой, второй, к третьей, четвертой и к пятой, шестой скобкам, соответственно.

$$\bar{D} \vee \big((A \vee B) \wedge (A \vee \bar{B}) \wedge (\bar{A} \vee B)\big)$$

Применим закон склеивания к первой и второй скобкам.

$$\bar{D} \vee \big(A \wedge (\bar{A} \vee B)\big)$$

Применим распределительный закон.

$$\bar{D} \vee (A \wedge \bar{A} \vee A \wedge B)$$

Применим закон противоречия.

$$\bar{D} \vee (0 \vee A \wedge B)$$

Применим закон исключения констант.

$$\bar{D} \vee A \wedge B$$

Запишем результат в соответствии с требованиями: not D or A and B

4. Кодирование информации. Алгоритмы обработки кодированной информации (2 балла) [Очень длинная строка]

Последовательности символов строятся по следующим правилам:

Исходная последовательность состоит из двух букв: “AA”. Далее на каждом шаге берется очередная буква английского алфавита, и пара таких букв добавляется в начало последовательности и после каждой буквы последовательности, полученной на предыдущем шаге. Например, последовательности, получаемые после первых 3 шагов, будут выглядеть таким образом (третья последовательность разбита на две строки для лучшего восприятия):

1. BBABBABB
2. CCBCCBCCACCBCCBCCACCBCCBCC
3. DDCDDCDDBDDCDDCDDBDDCDDCDDADDCDDCDDBDDCDDCDDBDDCDDCDDADDCDDCDDBDDCDDCD DBDDCDDCDD

Построение завершается, когда в результате очередного шага в последовательности впервые появляются буквы Z. Эта последовательность является результирующей.

Определите, какие буквы стоят в результирующей последовательности на позициях 961 376 769 и 1 035 574 967 097, считая от начала строки с 1. В ответе укажите их подряд в указанном порядке.

Ответ: КС

Решение:

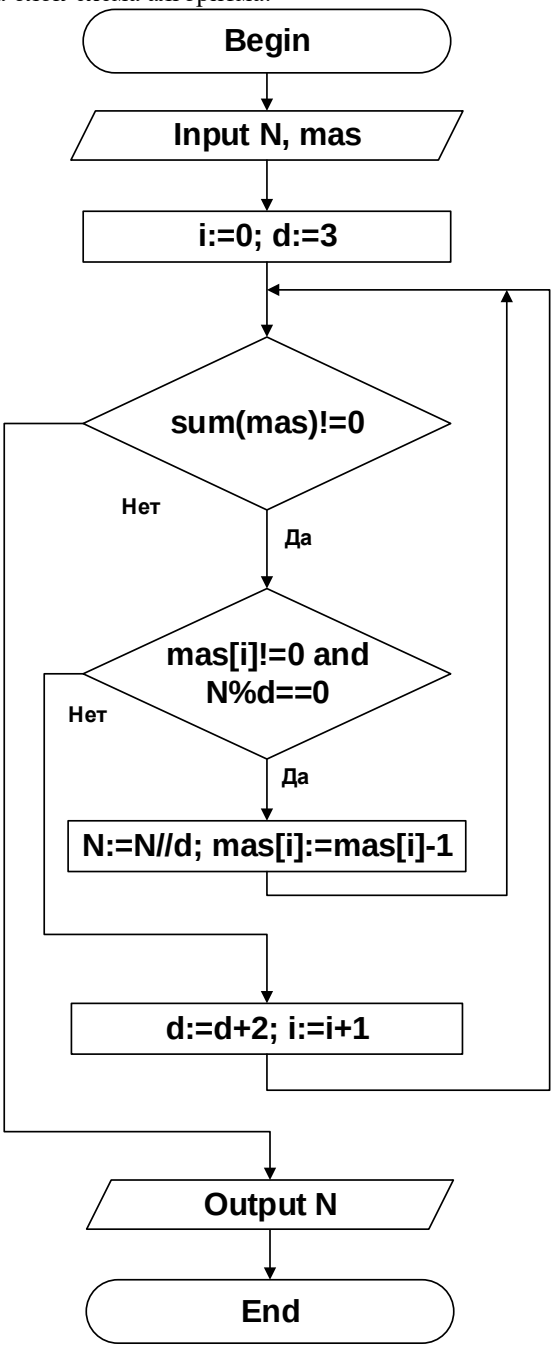
Обратим внимание, что в строке, в которой впервые появляется очередная буква алфавита, эта буква стоит на всех позициях, кроме тех, номер которых кратен 3 (считая с 1 от начала строки). А на позициях, кратных 3 стоят буквы, которые встречались в предыдущих строках. Также заметим, что при переходе к очередной строке позиция каждой буквы из предыдущей строки увеличивается в 3 раза. Так, буква B, которая встретилась впервые на второй позиции в строке «1», в строке «2» будет на шестой позиции, в строке «3» - на восемнадцатой позиции и т.д.

Строка с буквой Z будет 25-ой по счету. Рассмотрим число 961 376 769. Легко убедиться, что оно кратно 3, следовательно, это не буква Z. Если мы его поделим на 3, то получим 320 458 923 – это число также кратно 3, следовательно, в строке, в которой впервые появился символ Y, искомая буква стоит на позиции, кратной 3, то есть это не Y. Представим анализируемое число, как $k*3^n$, где k – сомножитель, не кратный 3. Получится, что 961 376 769 = $67*3^{15}$. Отсчитаем от Y в обратном порядке 15-ую букву – это буква «К». Мы определили первую часть ответа. Аналогично, 1 035 574 967 097 = $11*3^{23}$. Отсчитаем в обратном порядке от Y 23-ю букву и получим вторую часть ответа – «С».

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (1 балл)

[Три порога]

Дана блок-схема алгоритма.



На вход алгоритму подается целое положительное число N и массив из трех элементов [2, 3, 1].

Известно, что исполнение алгоритма успешно завершилось и было выведено число 1. Определите значение N, которое подали на вход алгоритма.

В ответе укажите целое число. Если такого числа не существует, запишите в ответ NULL. Если таких чисел несколько, запишите минимальное из них.

Примечания:

1. Индексация элементов массива начинается с 0.
2. Функция sum(mas) вычисляет сумму элементов массива.
3. Операция сравнения != обозначает «не равно».
4. Операция A//B означает вычисление частного при целочисленном делении A на B, а операция A%B означает вычисление остатка при целочисленном делении A на B.

Ответ: 7875

Решение:

Обратим внимание, что условием успешного завершения алгоритма являются нулевые значения всех элементов массива mas, поскольку сумма элементов должна быть равна нулю, исходно элементы имеют положительные значения, а уменьшение значения происходит только, если элемент еще не равен нулю и не более, чем на 1 за шаг цикла.

Значит, при успешном завершении алгоритма, в момент вывода N все элементы массива mas равны нулю. Также заметим, что альтернативой может быть аварийное завершение при выходе за границу массива, поскольку значение переменной i может превысить 2, а проверок на это в алгоритме не предусмотрено.

Уменьшение значения элемента массива mas происходит только сразу после деления числа N на d. Причем, для элемента mas[0] уменьшение связано на деление на d=3, для элемента mas[1] с делением на d=3+2=5, а для элемента mas[2] с делением на d=5+2=7. Таким образом, получается, что число N поделил на 3 два раза (исходное значение mas[0]=2), затем на 5 поделили 3 раза и, наконец, на 7 поделили еще один раз. И в результате получили 1. Значит, $N=1*3^2*5^3*7^1=7875$.

6. Телекоммуникационные технологии (2 балла). [Туда и обратно]

Маршрутизаторы (аппаратные или программные) выполняют задачу выбора оптимального маршрута следования IP пакета и его отправки по этому маршруту. Для принятия решения анализируется адрес получателя и устанавливается маршрут следования на основе таблиц маршрутизации.

В таблице маршрутизации присутствуют как минимум следующие поля:

1. адрес назначения (адрес IP-сети или IP адрес хоста) и маска назначения (она может задаваться или в десятичном формате, или в виде количества бит =1, то есть бит под адрес сети).
2. идентификатор порта, через который пакет идет до сети назначения (порт обозначается IP-адресом или внутренним номером),
3. шлюз (IP адрес, принадлежащий к одной из локальных сетей, непосредственно подключенных к маршрутизатору, на который необходимо отправить пакет, после того как пакет покинет порт, чаще всего – это адрес принадлежит следующему по маршруту маршрутизатору),
4. метрика (показатель качества маршрута).

На каждом маршрутизаторе сети присутствует таблица, описывающая структуру всей сети и иногда содержащая записи о маршрутах по умолчанию.

Маршрутные записи на сеть в качестве адреса (поля 1) содержат IP адрес сети и маску сети.

Маршрутные записи на хост (например, на один компьютер с известным IP адресом) в качестве адреса в первом и втором поле содержат IP адрес целевого хоста и маску, равную 255.255.255.255 (/32).

Запись по умолчанию отличается тем, что в первом поле адрес назначения и маска назначения имеют значения = 0.0.0.0 (0.0.0.0/0).

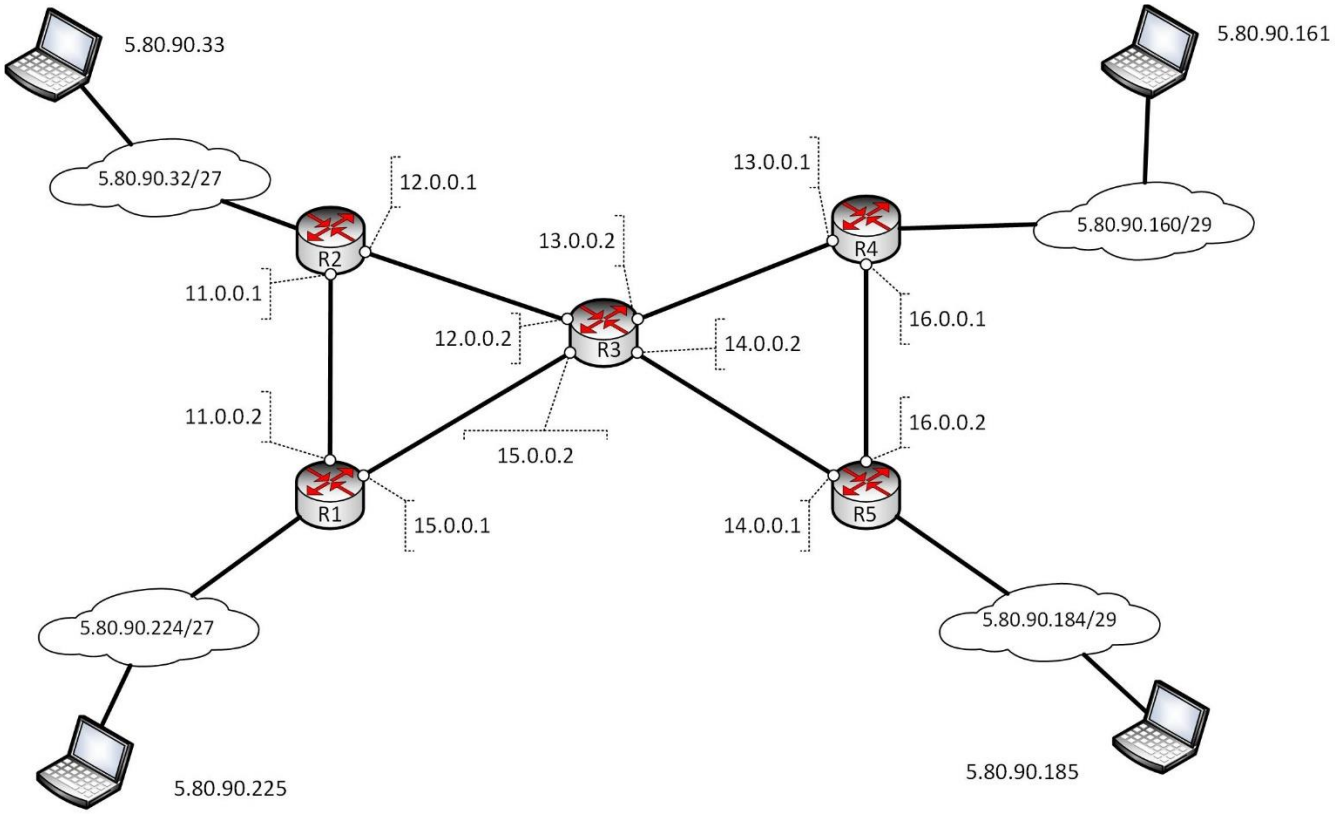
Если маршрутизатор получает пакет, адрес назначения которого, принадлежит сети, непосредственно подключенной к маршрутизатору, он направляет пакет соответствующему узлу, а в противном случае определяет дальнейший маршрут для этого пакета.

Когда маршрутизатору необходимо определить маршрут для продвижения IP пакета, то сначала по заголовку IP пакета определяется адрес назначения, а потом ищется подходящая запись в таблице маршрутизации. Несколько упрощая, можно считать, что маршрут ищется по принципу от частного к общему, т.е. сначала ищется маршрут на хост, потом маршрут на IP сеть, к которой принадлежит целевой адрес с маской /30 (255.255.255.252), потом с маской /29 (255.255.255.248) и т.д. Последним используется маршрут-по умолчанию.

Если обнаружено два формально подходящих маршрута с одинаковой маской, то выбирается тот из них, у которого метрика меньше.

Заметим, что маршрутизатор может «не знать» к IP-сети какого размера реально принадлежит адрес. Выбирается просто маршрут с подходящим адресом из имеющихся в таблице.

На рисунке приведена схема сети, где указаны адреса IP-сетей (в облачках) и отдельные адреса портов маршрутизаторов или компьютера (в пунктирных сносках). Приведены фрагменты таблиц маршрутизации каждого из маршрутизаторов.



R1			
Адрес назначения	Порт	Шлюз	Метрика
5.80.90.120/29	11.0.0.2	11.0.0.1	2
5.80.90.128/26	15.0.0.1	15.0.0.2	10
5.80.90.192/27	11.0.0.2	11.0.0.1	2
0.0.0.0/0	11.0.0.2	11.0.0.1	6

R2			
Адрес назначения	Порт	Шлюз	Метрика
5.80.90.128/25	12.0.0.1	12.0.0.2	1
5.80.90.184/29	12.0.0.1	12.0.0.2	2
5.80.90.224/27	11.0.0.1	11.0.0.2	3
0.0.0.0/0	11.0.0.1	11.0.0.2	1

R3			
Адрес назначения	Порт	Шлюз	Метрика
5.80.90.160/27	14.0.0.2	14.0.0.1	2
5.80.90.160/27	13.0.0.2	13.0.0.1	4
5.80.90.32/27	12.0.0.2	12.0.0.1	1
5.80.90.192/26	15.0.0.2	15.0.0.1	2

R4			
Адрес назначения	Порт	Шлюз	Метрика
5.80.90.0/26	13.0.0.1	13.0.0.2	5
5.80.90.96/29	13.0.0.1	13.0.0.2	2
5.80.90.192/27	16.0.0.1	16.0.0.2	2
0.0.0.0/0	13.0.0.1	13.0.0.2	1

R5			
Адрес назначения	Порт	Шлюз	Метрика
5.80.90.0/24	14.0.0.1	14.0.0.2	7
5.80.90.32/27	14.0.0.1	14.0.0.2	2
5.80.90.33/32	16.0.0.2	16.0.0.1	1
5.80.90.128/26	16.0.0.2	16.0.0.1	3

Выполняется обращение с хоста 5.80.90.225 на хост 5.80.90.161 и ответ в обратную сторону. Определите маршрут прохождения сетевого пакета туда и обратно. В ответе укажите номера (цифры) всех использованных маршрутизаторов для передачи пакета через запятую.

Например, если передача между двумя компьютерами, от IP1 к IP2, осуществлялась через маршрутизаторы R10, R11 туда и R11, R13, R10 обратно, то в ответе будет 10,11,11,13,10. Обратите внимание, что когда пакет идёт туда, адрес назначения в его заголовке – IP2, а когда ответ идёт обратно, адрес назначения будет IP1.

При решении задачи следует считать, что имеется вся другая необходимая конфигурационная информация (шлюзы по умолчанию на компьютерах, маршруты к непосредственно подключенным сетям и т.п.).

Ответ: 1,3,5,4,4,3,1

Решение:

Определим маршрут "туда". Адрес назначения анализируемого пакета 5.80.90.161. Представим для удобства в двоичном виде его последний октет (заметим, что первые 3 октета совпадают во всех адресах всех таблиц маршрутизаторов) - 10100001.

После отправки от узла 5.80.90.225 пакет попадет в маршрутизатор R1. На этом маршрутизаторе записаны 3 возможных маршрута + маршрут по умолчанию. Для каждого поля "Адрес назначения" в таблице маршрутизации запишем его с последним октетом в двоичном виде, указав символом | разделение адреса маской на адрес сети и адрес узла.

5.80.90.120/29 = ...01111|000
5.80.90.128/26 = ...10|000000
5.80.90.192/27 = ...110|00000

Для первой и третей записи видно, что пакет не принадлежит указанной сети назначения, а вот вторая подходит, поэтому пакет уйдет по маршруту 2 на порт 15.0.0.1 и, следовательно, попадет в маршрутизатор R3.

Действуя аналогично, представим в удобном виде таблицу маршрутизации маршрутизатора R3:

5.80.90.160/27 = ...101|00000
5.80.90.160/27 = ...101|00000
5.80.90.32/27 = ...001|00000
5.80.90.192/26 = ...11|000000

С точки зрения адреса назначения нам подойдут два первых маршрута, размеры сетей у них получаются одинаковыми, поэтому мы выбираем маршрут с меньшей метрикой, то есть пакет уходит в порт 14.0.0.2 и попадает в маршрутизатор R5.

Проанализируем таблицу маршрутизации R5.

5.80.90.0/24 = ...|00000000
5.80.90.32/27 = ...001|00000
5.80.90.33/32 = ...00100001
5.80.90.128/26 = ...10|000000

По адресу назначения нам подойдет только четвертый маршрут, следовательно, мы через порт 16.0.0.2 попадаем в маршрутизатор R4.

Маршрутизатор R4 непосредственно связан с сетью 5.80.90.160/29, которой принадлежит узел назначения и, следовательно, доставит пакет по назначению. Таким образом, маршрут "туда": R1-R3-R5-R4.

Аналогично построим обратный маршрут.

Наш новый адрес назначения 5.80.90.225. Его последний октет 11100001. Пакет из сети отправления попадет на маршрутизатор R4

Проанализируем таблицу маршрутизации R4.

5.80.90.0/26 = ...00|000000
5.80.90.96/29 = ...01100|000
5.80.90.192/27 = ...110|00000

Обратим внимание, что ни один из этих маршрутов не подойдет по адресу назначения. Следовательно, маршрутизатор R4 отправит пакет по маршруту по умолчанию, то есть, в порт 13.0.0.1. Пакет придет в маршрутизатор R3.

Его таблицу мы уже анализировали.

5.80.90.160/27 = ...101|00000
5.80.90.160/27 = ...101|00000
5.80.90.32/27 = ...001|00000
5.80.90.192/26 = ...11|000000

По адресу назначения нам подойдет только четвертый маршрут. Значит, пакет будет отправлен в порт 15.0.0.2 и попадет в маршрутизатор R1.

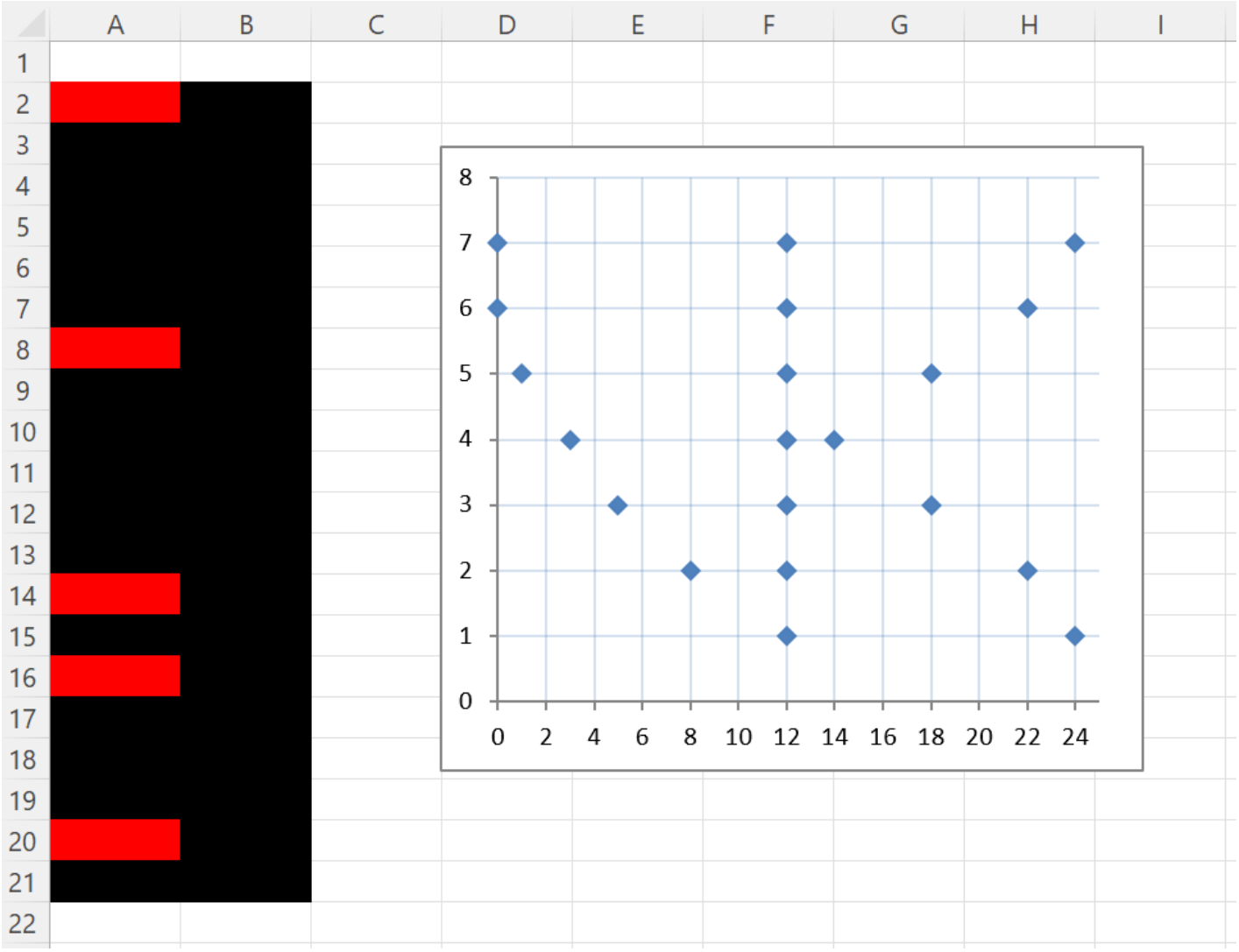
Этот маршрутизатор непосредственно связан с сетью 5.80.90.224/27, которой принадлежит узел назначения и доставит пакет этому узлу. Таким образом, обратный маршрут R4-R3-R1.

Тогда, ответ должен быть записан как 1,3,5,4,4,3,1.

7. Технологии обработки информации в электронных таблицах. Технологии сортировки и фильтрации данных (1 балл)

[VK]

Петя изучает различные виды диаграмм в электронных таблицах. Для того, чтобы потренироваться в построении точечной диаграммы с маркерами, Петя решил изобразить логотип IT-компании VK, заполнив целыми неотрицательными числами диапазон A2:B21:



Известно, что числа в столбце B отсортированы по невозрастанию. Какая минимальная сумма может быть у чисел в красных ячейках. В ответе укажите целое число.

Примечание. При построении точечной диаграммы с маркерами значения из левого столбца используются в качестве координат маркеров по оси абсцисс, а значения из правого столбца - для координат маркеров по оси ординат.

Ответ: 30

Решение:

Заметим, что на диаграмме ровно 20 маркеров, следовательно, каждому маркеру соответствует ровно одна пара значений в диапазоне A2:B21. Поскольку значения в столбце B (правом столбце) соответствуют значениям координат точек по оси ординат, легко построить значения этого столбца в указанном порядке сортировки. Сразу отметим позиции красных ячеек в соответствии с условием:

	7
	7
	7
	6
	6
	6
	5
	5
	5
	4
	4
	4
	3
	3
	3
	2
	2
	2
	1
	1

Следовательно, нам нужно одно минимальное значения для маркера с ординатой 7, одно минимальное значение для маркера с ординатой 5, два минимальных значения для маркера с ординатой 3 и одно минимальное значение для маркера с ординатой 1. Из графика можно получить, что это числа 0, 1, 5, 12, 12. Их сумма будет равна 30.

8. Технологии программирования (3 балла) [Система контроля версий]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

IT-компания «VK» имеет огромную инфраструктуру проектов, и чтобы разработчики могли гармонично работать вместе и структурировать вносимые в код изменения, им необходима система контроля версий (VCS).

Полноценные системы контроля версий устроены достаточно сложно, и некоторые компании даже разрабатывают свои собственные VCS, заточенные под конкретные нужды или особенности рабочего процесса. Разумеется, разработчики из «VK» могут справиться с задачей реализовать свою VCS, но сегодня эта задача предлагается вам.

Ваша примитивная система контроля версий должна иметь вид ацикличного ориентированного графа изменений состояния проекта.

«Исток» графа — стартовая вершина, соответствующая изначальному состоянию проекта. Это единственная вершина, в которую не входят ребра.

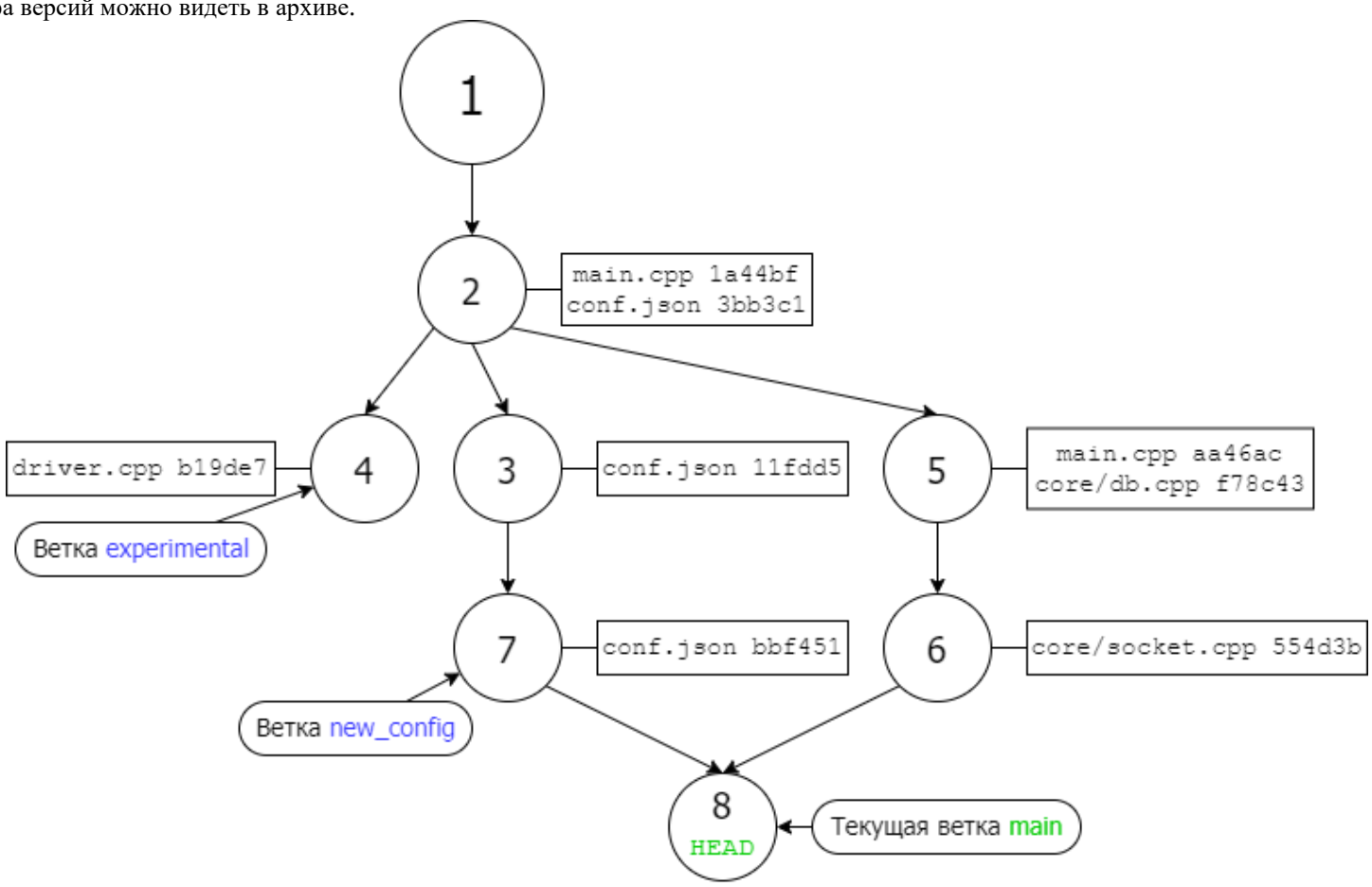
Каждая вершина, кроме корня, соответствует определенному коммиту. Коммит — блок из одного или более изменений.

Версия проекта, задаваемая вершиной — набор всех изменений на путях между стартовой вершиной и данной.

Есть несколько выделенных веток, каждая имеет уникальное имя и задается указателем на определенную вершину графа. С веткой ассоциируется версия проекта, соответствующая вершине, на которую она указывает.

Из всех веток выделяется текущая ветка — версия проекта, с которой сейчас работает пользователь. Вершину, на которую указывает текущая ветка, будем обозначать как HEAD.

Пример графа можно видеть ниже. Рядом с каждым коммитом указаны соответствующие ему изменения. Вершина номер 88 соответствует команде merge между ветками «main» и «new_config» (описание команд см. ниже). Набор команд, позволяющий получить приведенную структуру графа версий, приведен в первом тесте. Пошаговые иллюстрации изменения графа версий можно видеть в архиве.



Изначально граф состоит из единственной вершины с номером 11, на которую указывает текущая ветка «main». Требуется поддерживать следующие команды:

«add <файл> <хеш изменений>» — запомнить изменения, внесенные в данный файл. Хеш однозначно описывает набор изменений в файле. Иными словами, хеши двух независимых изменений совпадают тогда и только тогда, когда состояния файла до и после изменения одинаковы.

Иными словами, если в пустой файл добавляется строчка «print(something)», и в непустой файл добавляется та же строчка, хеши этих двух изменений будут различными.

«commit» — подвесить к HEAD новую вершину, состоящую из всех изменений (add), сделанных с момента предыдущего успешного коммита. Новой вершине присваивается первый неиспользованный натуральный номер, после чего HEADHEAD перемещается на нее.

Операция возможна только тогда, когда множество сделанных изменений непустое. В противном случае требуется выдать ошибку «ERROR: no changes», при этом новая вершина не создается.

«reset <номер>» — переместить HEAD на вершину с данным номером. Гарантируется, что вершина с данным номером существует. Если присутствуют несохраненные (commit) изменения (add), операция отклоняется с ошибкой «ERROR: uncommitted changes».

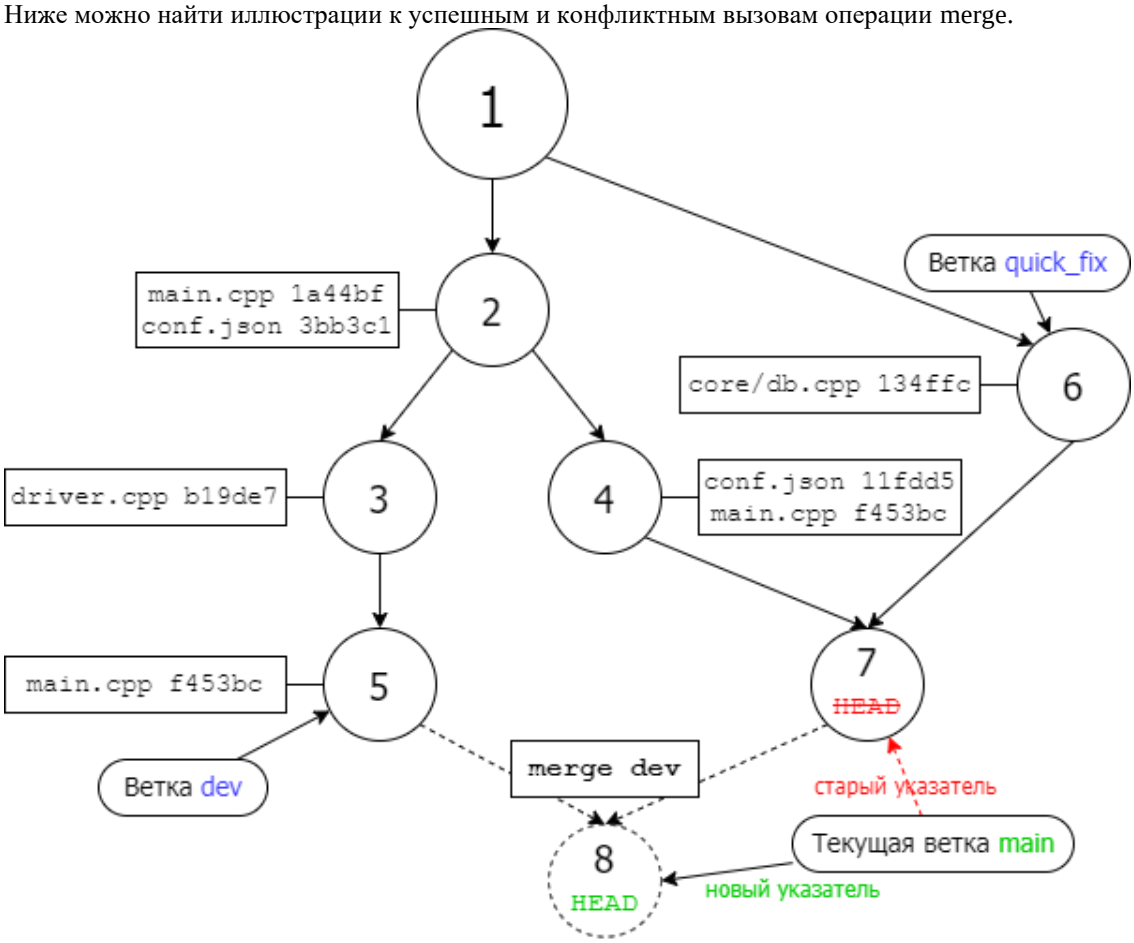
«checkout <имя ветки>» — поменять текущую ветку на ветку с указанным именем (и, соответственно, переместить HEAD на версию, соответствующую выбранной ветке). Если ветки с таким именем не существует, создать новую ветку с таким именем, которая будет указывать на HEAD, после чего сделать ее текущей. Если присутствуют несохраненные изменения, операция отклоняется с ошибкой «ERROR: uncommitted changes», аналогично операции reset.

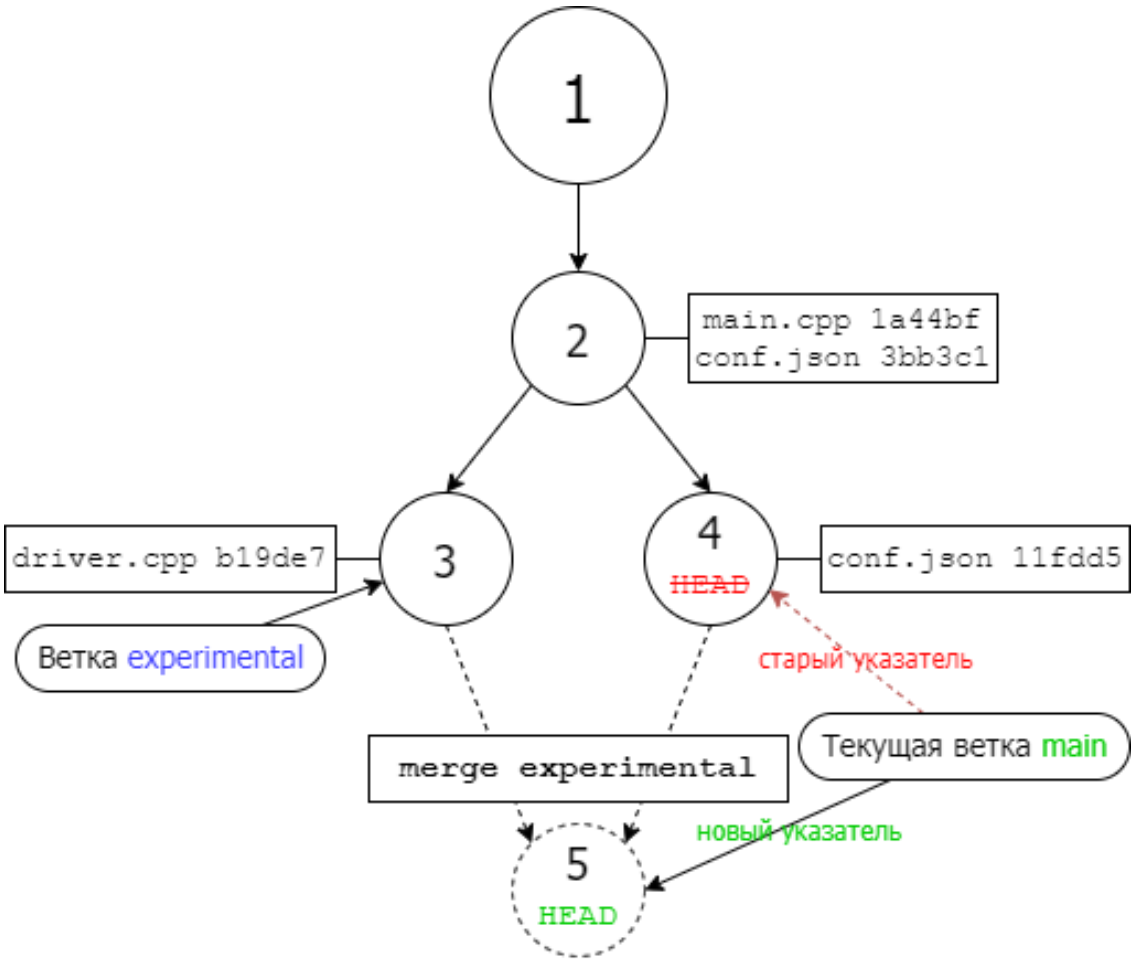
«merge <имя ветки>» — объединить изменения в текущей ветке и в ветке с указанным именем. При этом создается новая вершина, которая подвешивается к HEAD и к вершине, соответствующей второй ветке. HEAD при этом перемещается на новую вершину, а указатель второй ветки не двигается. Гарантируется, что имя второй ветки не совпадает с текущей. Если есть незакоммиченные изменения, операция отклоняется с ошибкой «ERROR: uncommitted changes». Если несохраненных изменений нет, проверяется отсутствие конфликтов при объединении. Конфликтом считается ситуация, когда в состояниях проекта, соответствующим данным веткам, с одним и тем же файлом сделаны различные изменения. Иными словами, если обозначить множество изменений определенного файла в текущей ветке как D_1, а во второй — как D_2, то конфликт возникает, когда оба множества $D_1 \setminus D_2$ и $D_2 \setminus D_1$ непустые. В таком случае операция отклоняется с ошибкой «ERROR: conflicts detected».

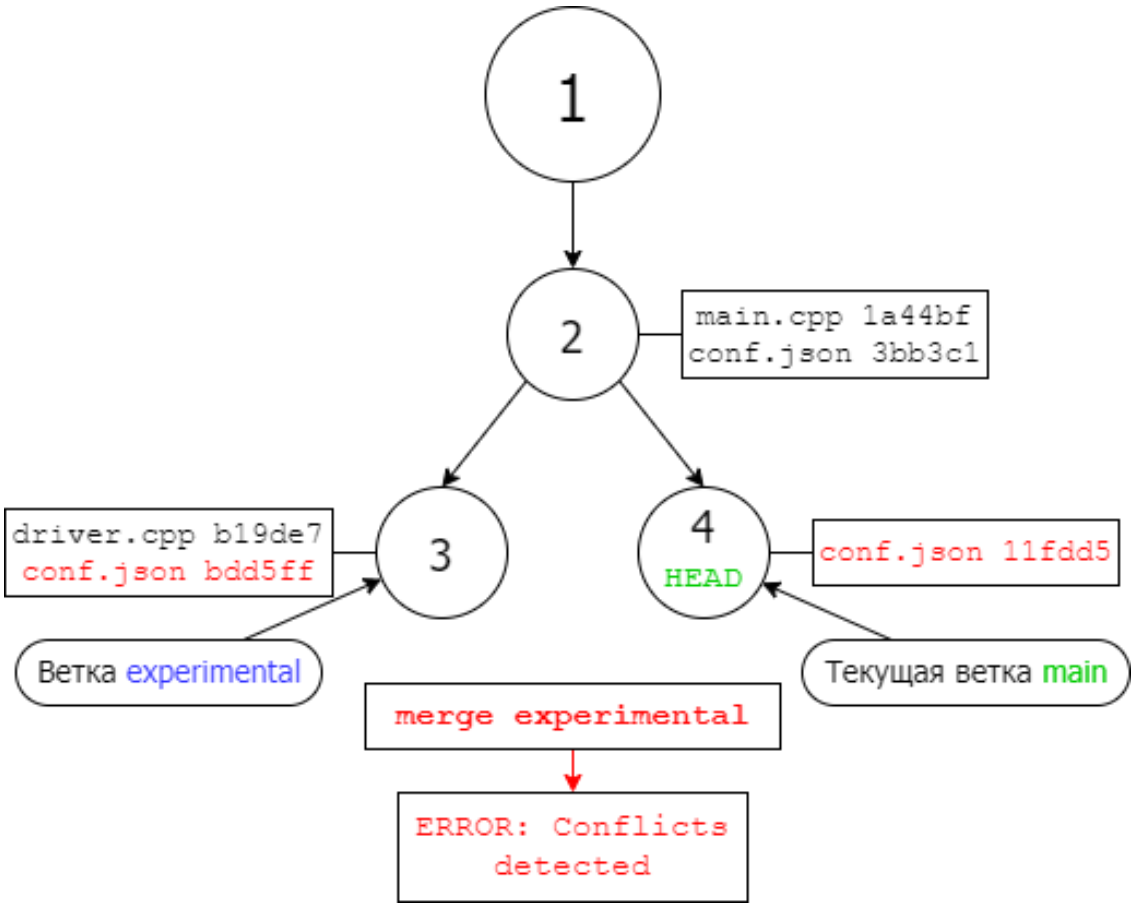
Обратите внимание, что множества коммитов в двух версиях сравниваются как математические множества. Стоит считать, что ситуаций, когда в разных ветках одни и те же изменения одного файла располагаются в разном порядке, не существует.

Если какое-то изменение содержится в версиях разное ненулевое количество раз (например, изменение может присутствовать дважды после merge двух веток, в которые оно входило), конфликт не возникает. Конфликт возникает только если в каждой версии есть изменение одного и того же файла, которого нет в другой версии.

Ниже можно найти иллюстрации к успешным и конфликтным вызовам операции merge.







Вам дается последовательность команд, которые требуется обработать. Для каждой команды выведите через пробел слово «OK» и номер вершины, на которую указывает HEAD, если команда выполнена успешно, или же соответствующую ошибку, если команда отклонена.

Формат входных данных

В первой строке дано единственное целое число T — количество наборов входных данных, которые вам предстоит обработать (1≤T≤20). Далее следуют описания наборов входных данных.

Каждый набор входных данных начинается со строки, содержащей единственное целое число n — количество команд в наборе (0≤n≤400). В i-й из следующих n строк задается i-я команда.

Гарантируется, что имена веток состоят только из маленьких латинских букв ('a' — 'z') и нижних подчеркиваний ('_'), а хеши изменений — уникальные шестнадцатеричные строки длины ровно 6 (состоят из цифр и маленьких латинских букв от 'a' до 'f'). Имена файлов в команде add состоят из маленьких латинских букв, точек и слешей ('/').

Также гарантируется, что команде reset всегда передается существующая вершина, а команде merge — существующая ветка, не совпадающая с текущей.

Формат выходных данных

Для каждого набора входных данных в порядке их следования во вводе сначала выведите строку «Test case <номер>» (наборы нумеруются от 1 до T), а затем n результатов выполнения команд, каждый в своей строке.

Результат выполнения каждой команды должен соответствовать либо формату «OK <HEAD>», либо формату «ERROR: <сообщение об ошибке>».

Пример

Стандартный ввод	**Стандартный вывод**
2 12 add a.cpp 1f1f1f reset 1 commit checkout dev reset 1 add a.cpp d2d2d2 commit merge main reset 1 add b.cpp abc123 commit merge main 23 add main.cpp 1a44bf add conf.json 3bb3c1 commit add conf.json 11fdd5 checkout new_config commit checkout new_config checkout main reset 2 checkout experimental add driver.cpp b19de7 commit checkout main add main.cpp aa46ac add core/db.cpp f78c43 commit add core/socket.cpp 554d3b commit checkout new_config add conf.json bbf451 commit checkout main merge new_config	 Test case 1 OK 1 ERROR: uncommitted changes OK 2 OK 2 OK 1 OK 1 OK 3 ERROR: conflicts detected OK 1 OK 1 OK 4 OK 5 Test case 2 OK 1 OK 1 OK 2 OK 2 ERROR: uncommitted changes OK 3 OK 3 OK 3 OK 2 OK 2 OK 2 OK 4 OK 2 OK 2 OK 2 OK 5 OK 5 OK 6 OK 3 OK 3 OK 7 OK 6 OK 8

Примечание

Пустая строка в выводе в примере добавлена специально, чтобы результаты выполнения команд находились ровно напротив команд, которым они соответствуют. Ваше решение не должно выводить пустую первую строку.

Решение:

Требовалось аккуратно реализовать требуемые команды. Посмотрим, что нам нужно хранить, чтобы корректно выполнять их:

• changes — множество изменений, ожидающих команды commit;

• структуру графа версий;

• data — набор изменений, соответствующих каждой вершине графа;

• branches — номера вершин, на которые ссылаются все существующие ветки;

• head_branch — имя текущей ветки.

Будем хранить изменения для каждого файла отдельно. Хеш изменения можно было воспринимать как строку или как шестнадцатеричное число, будем считать хеш строкой. Тогда будем:

• хранить changes как dict[str, set[str]], где changes[file] — множество изменений, примененных к файлу file;

• хранить data как list[dict[str, set[str]]], где data[v] — множество изменений, соответствующих вершине v, в том же формате;

• хранить branches как dict[str, int];

• воспринимать HEAD как сокращенное обозначение branches[head_branch] (в том числе и на присвоение, присвоение HEAD = x означает присвоение в branches по ключу head_branch).

Хранить граф можно было несколькими способами. Будем хранить для каждой вершины сразу всех предков в множестве (первый вариант) или массиве (второй вариант) ancestors[v].

Не будем приводить реализацию проверки простых условий, вроде отсутствия незакоммиченных изменений. Приведем описание реализаций простых команд, после чего опишем реализацию commit, merge.

1. Команда add добавляет новую запись в changes как changes[file].insert(hash).

2. Команда reset на вершину v выполняет присвоение branches[head_branch] = v.

3. Команда checkout проверяет, существует ли переданная ветка branch, и если нет, то присваивает branches[branch] = HEAD, после чего всегда меняет head_branch на branch.

Команда commit создает новую вершину, записывает текущий HEAD ее родителем и присваивает в data[v] данные из changes. После чего changes очищается, а HEAD двигается на новую вершину. Ниже приведен псевдокод для второго варианта, в первом вместо + [v] будет | {v}:

```
v <- свободный номер
data[v] = copy(changes)
ancestors[v] = ancestors[HEAD] + [v]
changes.clear() HEAD = v
```

Далее приведем реализацию merge. Для операции требовалось сначала проверить отсутствие конфликтов. Для этого необходимо найти множества изменений каждого файла среди всех предков двух веток и сравнить. Пройдемся по всем предкам и объединим множества изменений для каждого файла.

```
log = dict()
for node in ancestors[v]:
    for file in data[node]:
            if file not in log: log[file] = set()
            log[file] |= data[node][file]
```

В конце получим current_log и side_log — изменения в текущей и в побочной ветках. Чтобы не было конфликта, надо проверить, что для каждого файла множество изменений в одной из веток полностью лежит в другой, то есть что размер current_log[file] & side_log[file] равен минимуму размеров current_log[file] и side_log[file].

Для merge после этого надо просто создать новую вершину и подвесить к вершинам текущей и побочной веток. При этом заполнить ancestors[v] как ancestors[HEAD] | ancestors[branches[branch]] | {v}.

9. Технологии программирования (4 балла) [Парковка]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	3 секунды
Ограничение по памяти	256 мегабайт

Специально для n сотрудников ИТМО, пользующихся личными автомобилями, планируется открыть парковку. На парковке должно быть ровно n парковочных мест, каждому сотруднику должно достаться свое место.

Для экономии мест парковка будет разбита на несколько «рядов». Места в каждом ряду нумеруются от 1 (самое дальнее от въезда) до длины ряда (самое ближнее ко въезду), и дальние места недоступны, пока не освободятся все более ближние.

Для каждого сотрудника известно, в какое время он приезжает, и в какое время заканчивает работу. Так как сотрудники ИТМО — очень трудолюбивые люди, каждый из них приезжает на работу в один день, а уезжает уже в следующий. Для каждого известно время, в которое он приезжает на работу t_i^{in}, и время, в которое он уезжает на следующий день t_i^{out}. Требуется назначить места сотрудникам так, чтобы никому из них не понадобилось ждать

- появления доступного парковочного места, когда он приезжает;
- возможности выехать, когда он заканчивает работу.

Более формально, если сотрудникам i и j назначены места p_i и p_j в одном ряду, и $p_i<p_j$, должно выполняться $t_i^{in}<t_j^{in}$ и $t_i^{out}>t_j^{out}$.

Определите, какое минимальное число рядов понадобится, чтобы можно было распределить всех сотрудников по местам на парковке указанным образом, и найдите соответствующее распределение сотрудников по местам.

Формат входных данных

В первой строке ввода дано единственное целое число T — количество наборов входных данных ($1\le T\le 100$). Далее следуют описания наборов входных данных.

В первой строке описания набора входных данных дано единственное целое число n — количество сотрудников, которых необходимо разместить на парковке.

Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^5.

Во второй строке набора входных данных через пробел перечислены n целых чисел t_i^{in} — времена приезда сотрудников на работу ($1 \le t_i^{in} \le 10^9$). В третьей строке в том же формате перечислены n целых чисел $t_i out$ — времена отъезда сотрудников ($1 \le t_i^{out} \le 10^9$).

Формат выходных данных

Выведите ответ на задачу для каждого набора входных данных в том порядке, в котором они перечислены во вводе.

В первой строке ответа выведите единственное целое число k — минимальное необходимое количество рядов. В i-й из следующих k строк выведите описание i-го ряда: первое число в строке cnt_i должно быть равно количеству мест в ряду, после чего должны следовать cnt_i целых чисел от 1 до n — номера сотрудников, занимающих места этого ряда, в порядке от самого глубокого к самому ближнему ко въезду.

Если существует несколько различных ответов, минимизирующих k, выведите любой из них.

Пример

Стандартный ввод	**Стандартный вывод**
4 2 1 2 3 4 3 5 7 6 4 3 1 4 1 2 3 4 8 7 5 6 2 3 1 2 5	2 1 1 1 2 2 2 1 3 1 2 2 3 1 2 3 1 4 1 2 2 1

Решение

Для решения задачи надо заметить следующий факт: последовательность сотрудников может быть расположена в один ряд, если их интервалы времени $[t_i^{in}, t_i^{out}]$ последовательно вложены друг в друга (строго вложены в первом варианте и нестрого во втором). Также можно было бы расположить двух людей в одном ряду, если их интервалы времени не пересекаются, но поскольку все приезжают в первый день, а уезжают во второй, такая ситуация невозможна.

Таким образом, надо разбить всех n сотрудников на минимальное число последовательностей со вложенными отрезками времени между прибытием и отъездом. Отсортируем всех сотрудников по неубыванию времени приезда. Тогда если $i < j$ и $t_i^{out} > t_i^{out}$, их можно последовательно поставить в один ряд.

Здесь стоит заметить, что чтобы решение работало корректно, люди с одинаковым t^{in} должны быть упорядочены по неубыванию t^{out} (иначе указанное выше условие неверно).

Итак, исходная задача превратилась в задачу разбиения полученного массива t^{out} на как можно меньшее число убывающих подпоследовательностей. Конструктивно можно показать, что на самом деле минимальное количество последовательностей в разбиении на убывающие равно длине наибольшей возрастающей подпоследовательности, и решение похоже на быстрый алгоритм поиска НВП.

Пройдемся по массиву слева-направо и будем поддерживать минимальный по размеру набор рядов, необходимый для размещения первых i сотрудников, а среди всех таких — лексикографически максимизирующий последние t^{out} в рядах. Ряды при этом будем хранить в отсортированном по t^{out} последнего сотрудника порядке. Пусть p_i — этот самый последний t^{out} в i-м ряду, и $p_1 \le p_2 \le \ldots \le p_k$. Рассмотрим очередного сотрудника с временем выезда t.

1. Если существует $p_i > t$, то можно поставить нового сотрудника в конец i-го ряда. Получится последовательность времен выезда $[p_1, \ldots, p_{i-1}, t, p_{i+1}, \ldots, p_k]$. Чтобы она получилась лексикографически максимальной после пересортировки, надо поставить t после минимально возможного p_i . Найти $\min p_i > t$ можно с помощью бинпоиска за время O(log n). Более того, можно заметить, что если он минимальный, то $p_{i-1} \le t < p_i \le p_{i+1}$, поэтому ряд $[p_1, \ldots, p_{i-1}, t, p_{i+1}, \ldots, p_k]$ сразу будет упорядочен и его не придется пересортировывать.

2. Если не существует такого p_i , то надо создать новый ряд для этого сотрудника и сделать $p_{k+1} = t$. Можно показать, что если заранее оставить свободным место для этого сотрудника в каком-то ряду, то придется создать этот новый ряд раньше, и ответ получится не меньше, чем при таком алгоритме. В конце будет достаточно пройтись по собранным рядам и вывести их в требуемом формате. Более формально доказать оптимальность алгоритма можно, заметив, что p_i совпадают со значениями динамического программирования для поиска НВП, поэтому количество рядов будет в точности равно длине НВП, а получить разбиение меньшего размера нельзя (например, по теореме Дилуорса).

Отборочный этап. Первый тур (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла) [Теорема Никомаха]

У Пети появился личный суперкомпьютер. Он долго думал, какой бы задачей его нагрузить и написал программу, которая подсчитывает следующую сумму чисел, записанных в позиционных системах счисления:

$R_{10}=1000_2 + 1000_3 + 1000_4 + \ldots + 1000_{N-1} + 1000_N + 1$

С помощью своей программы Петя сумел за несколько минут найти значение R_{10} для $N=16^9{}_{10}$.

Петя решил подразнить своего друга Васю, зная, что у того только старенький ноутбук и предложил Васе решить эту задачу на своём ноутбуке хотя бы за 3 часа. Через 3 часа Петя зашел к Васе и с удивлением увидел, что Вася тоже получил правильный ответ. Петя заметил, что у Васи остался открытым браузер и в поисковой строке набрано «Теорема Никомаха».

Найдите и Вы число R_{10}, которое получили Петя и Вася и запишите в ответ сумму цифр этого числа.

Примечание. Если у Вас нет суперкомпьютера, Вы тоже можете воспользоваться поиском информации в сети Интернет.

Ответ: 208

2. Кодирование информации. Системы счисления (1 балл) [Единицы]

Сколько существует положительных вещественных чисел, меньших $0,25_{10}$ таких, что:

1. Если это число умножить на 2048_{10}, то результат будет целым числом.
2. Если записать это число в двоичной системе счисления, запись будет содержать не менее 6 единиц, идущих подряд.

Ответ: 20

3. Кодирование информации. Количество информации. (2 балла) [Автомобильные номера]

Петя попал в виртуальный мир. Жители этого мира используют алфавит из 32 букв и восьмеричную систему счисления. А еще, каждому жителю этого мира в день 22-летия дарят автомобиль. При этом автомобильный номер генерируется по следующим правилам:

1. Номер состоит из N идущих подряд букв алфавита и затем из K идущих подряд цифр (N>K).
2. Каждая буква выбирается случайным образом так, что с равной вероятностью может оказаться любым символом алфавита.
3. Каждая цифра выбирается случайным образом так, что с равной вероятностью может оказаться любой цифрой восьмеричной системы счисления.

Среди всех возможных номеров, которые могут быть получены таким способом, жители виртуального мира особо выделяют три вида номеров:

1. Серебряный номер – это такой номер, все цифры в котором одинаковые.
2. Золотой номер – это такой номер, все буквы в котором одинаковые.
3. Платиновый номер – это такой номер, в котором все буквы одинаковые и все цифры одинаковые.

Обратим внимание, что платиновый номер всегда одновременно также и золотой, и серебряный.

Известно, что сообщение, что житель получил автомобиль с золотым номером, несет в себе ровно на 9 бит больше информации, чем сообщение, что житель получил автомобиль с серебряным номером. А сообщение, что житель получил автомобиль с платиновым номером, несет в себе ровно 21 бит информации.

Определите значения N и K и укажите в ответ через пробел два числа в десятичной системе счисления: сначала значение N, а затем значение K.

Ответ: 4 3

4. Кодирование информации. Информационный объем (1 балл) [RLE 3]

Петя пишет программу для школьного проекта. Один из модулей программы должен сохранять в памяти растровое изображение, длина строки которого составляет X пикселей. При этом используется палитра из 256 цветов. Петя хорошо учится в школе, но пока знает только самый простой метод: сохранять каждую строку изображения как последовательность кодов цветов пикселей, используя для записи каждого кода одинаковое, минимально возможное для записи всех доступных в палитре цветов количество бит. Опытный Вася хочет помочь Пете. Он вспомнил задачи, которые решал на олимпиадах и решил использовать метод RLE (Run-length encoding). При использовании этого метода для кодирования строки растрового изображения, каждая подстрока идущих подряд пикселей с одинаковыми цветами, представляется как последовательность из двух значений A и B, где B – значение цвета пикселя, а A – количество повторений идущих подряд пикселей с цветом B. Даже если в строке встречается один неповторяющийся пиксель, он все равно будет представлен как два значения: «1B». При записи в память каждого кода значения A используется одинаковое, минимально возможное для записи всех возможных значений A количество бит. При записи в память каждого кода значения B используется одинаковое, минимально возможное для записи всех доступных в палитре цветов количество бит. Отметим, что количества бит, которые потребуются для хранения кодов A и B, зависят от количества пикселей в кодируемых строках и количества цветов в палитре изображения соответственно.

Известно, что некоторая строка изображения представляет собой ровно три идущих друг за другом последовательности из пикселей с одинаковыми цветами. При какой длине строки X выигрыш по памяти для её хранения в случае использования описанного выше метода RLE по отношению к методу, использованному Петей, составит ровно 217 бит? В ответе укажите целое число. Если таких значений несколько, укажите минимальное из них.

Ответ: 32

5. Основы логики. Анализ логических функций (2 балла) [Три следования]

Дано следующее логическое тождество:

$$\Big(\big(F(A,B,C) \to A \wedge F(A,B,C)\big) \to B \wedge F(A,B,C)\Big) \to C \wedge F(A,B,C) = \text{истина}$$

Найдите такую логическую функцию трех переменных F(A,B,C), для которой это тождество будет выполняться. Если таких логических функций несколько, выберите ту из них, в таблице истинности которой будет минимальное количество ложных значений. В ответе укажите логическую формулу, которая может содержать логические переменные A, B, C и не более чем три логические операции из набора {инверсия, конъюнкция, дизъюнкция}. Если таких функций нет, запишите в ответ NULL.

Комментарий по вводу ответа: операнды вводятся большими латинскими буквами; логические операции обозначаются, соответственно как not, and и or. Запись не должна содержать скобок. Пример записи ответа: A or not B

Ответ: C or not B and A || C or A and not B || not B and A or C || A and not B or C

6. Основы логики. Упрощение логического выражения (1 балл) [Перевертыши]

Упростите логическое выражение или укажите его результат (при его однозначности). Результат упрощения может содержать только операции инверсии, конъюнкции и дизъюнкции.

(((((A xor B) ↔ C) xor D) ↔ C) xor B) ↔ A

*Комментарий по вводу ответа: операнды вводятся большими **латинскими** буквами; логические операции обозначаются, соответственно как **not**, **and** и **or**.*

Скобки используются только для изменения порядка выполнения операций. Если порядок выполнения операций очевиден из их приоритетов – дополнительное использование скобок считается ошибкой.

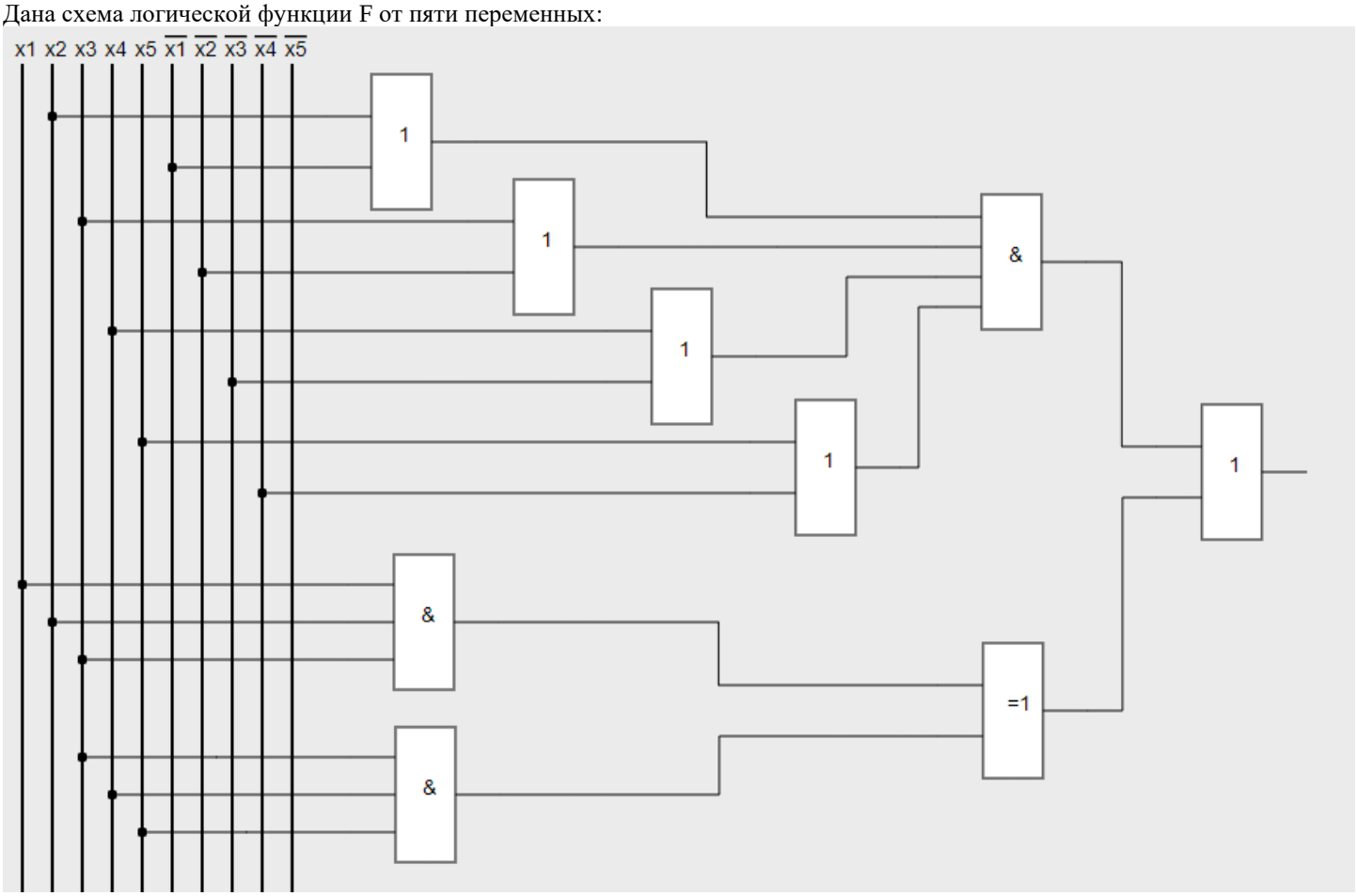
При однозначном ответе – истинный ответ обозначается как 1, а ложный как 0.

Пример записи ответа: (A or not B) and C

Ответ: not D

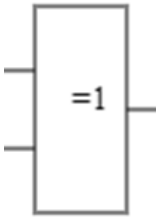
7. Основы логики. Синтез выражения по логической схеме (2 балла) [Две ветви]

Дана схема логической функции F от пяти переменных:



Сколько существует различных наборов значений переменных x1, x2, x3, x4, x5 таких, что в результате будет получено значение 1 (истина)? В ответе укажите число.

Примечание. На схеме использованы следующие обозначения логических операторов:

Конъюнкция	Дизъюнкция	Исключающее ИЛИ
		

Ответ: 10

8. Алгоритмизация и программирование. Формальный исполнитель (3 балла) [Волны]

Над символьной строкой последовательно в бесконечном цикле осуществляются следующие действия:

1. Заменить в строке все подстроки 'CC' на подстроку 'BCACB'.
2. Заменить в строке все подстроки 'BB' на подстроку 'ACBCA'.
3. Заменить в строке все подстроки 'AA' на подстроку 'CBABC'.

После выполнения третьего действия, опять выполняется первое и т.д. Каждое действие выполняется над строкой, получившейся в результате выполнения предыдущего действия.

После выполнения каждого действия для получившейся строки вычисляются четыре значения: общее количество символов в строке, количество символов 'A' в строке, количество символов 'B' в строке и количество символов 'C' в строке.

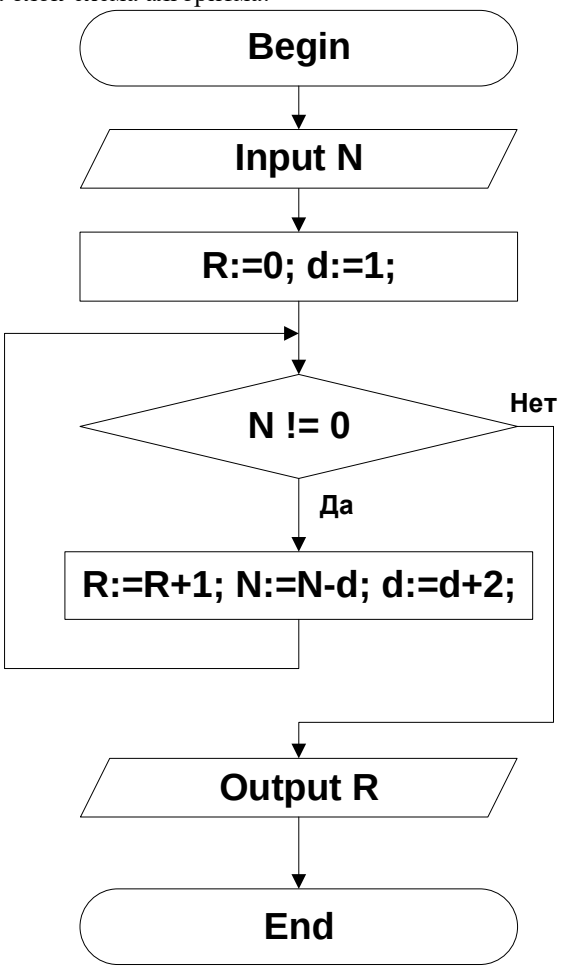
Пусть перед началом первого выполнения первого действия была строка 'ABCCBA'. Известно, что после выполнения некоторого действия в строке оказалось 80808021 символов. Определите для этой строки и запишите в ответ через пробел сначала количество символов 'A', затем количество символов 'B' и затем количество символов 'C'. Если такое состояние строки невозможно, запишите в ответе NULL.

Примечание. Поскольку в ответе будут получаться достаточно большие числа, рекомендуется внимательно проверить ввод ответа.

Ответ: 26936005 13468006 40404010

9. Алгоритмизация и программирование. Блок-схема, обратная задача (1 балл) [Почему нечетные?]

Дана блок-схема алгоритма:



Какое целое положительное число N нужно подать на вход, чтобы на выходе получилось значение R=975? В ответе укажите число.

Ответ: 950625

10. Алгоритмизация и программирование. Блок-схема, обратная задача (3 балла) [Спирали 2]

Петя совершенствует свою программу, которая по заданному числу N строит квадратную матрицу NxN, заполненную последовательно натуральными числами по спирали по часовой стрелке, начиная с верхнего левого угла.

Теперь его программа умеет также осуществлять циклический сдвиг элементов матрицы вдоль спирали против часовой стрелки на заданное количество шагов. Воспользовавшись программой, Петя получил такие примеры:

N=4, исходная матрица	N=4, сдвиг на 1 шаг
1 2 3 4 12 13 14 5 11 16 15 6 10 9 8 7	2 3 4 5 13 14 15 6 12 1 16 7 11 10 9 8
N=5, исходная матрица	N=5, сдвиг на 4 шага
1 2 3 4 5 16 17 18 19 6 15 24 25 20 7 14 23 22 21 8 13 12 11 10 9	5 6 7 8 9 20 21 22 23 10 19 3 4 24 11 18 2 1 25 12 17 16 15 14 13

Петя построил матрицу для N=15 и не поленился сделать сдвиги на все возможные значения количества шагов от 0 до N^2-1. Для каждого значения количества шагов циклического сдвига (K) Петя посчитал сумму элементов, расположенных на главной диагонали матрицы (от левого верхнего до правого нижнего угла матрицы). Он обнаружил, что при некотором значении K получается минимально возможная сумма. Найдите это значение K и сумму элементов на главной диагонали матрицы для этого K. В ответе через пробел запишите два числа: сначала значение K, а затем сумму элементов на главной диагонали матрицы после циклического сдвига на K шагов.

Ответ: 49 1303

Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Электронные таблицы. Адресация ячеек и вычисления (1 балл) [Числа Сабита]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D	
1		0	1	2	3
2		=3*СТЕПЕНЬ(2;B1)-1			
3		=ЧАСТНОЕ(B2;СТЕПЕНЬ(2;$A1))			
4					

Ячейку B2 скопировали во все ячейки диапазона B2:AJ2. Ячейку B3 скопировали во все ячейки диапазона B3:AJ3. В ячейку A3 поместили формулу

=СУММ(B3:AJ3)

Какое целое положительное число нужно записать в ячейку A1, чтобы в ячейке A3 получилось значение 3? В ответе укажите целое число. Если такого числа не существует, укажите в ответе NULL.

Ответ: 34

2. Электронные таблицы. Графики и диаграммы (2 балла) [Взлеты и падения]

Петя изучает числовые последовательности. В электронных таблицах он заполнил все ячейки первой строки, начиная со столбца B, последовательно натуральными числами.

В ячейку B2 Петя поместил формулу

=ЕСЛИ(ОСТАТ(B1;$A2)<$A2/2;ОСТАТ(B1;$A2);$A2-ОСТАТ(B1;$A2)),

а затем скопировал ячейку B2 во все ячейки второй строки, начиная с C2.

В ячейку B3 Петя поместил формулу

=ЕСЛИ(ОСТАТ(B1;$A3)<$A3/2;ОСТАТ(B1;$A3);$A3-ОСТАТ(B1;$A3)),

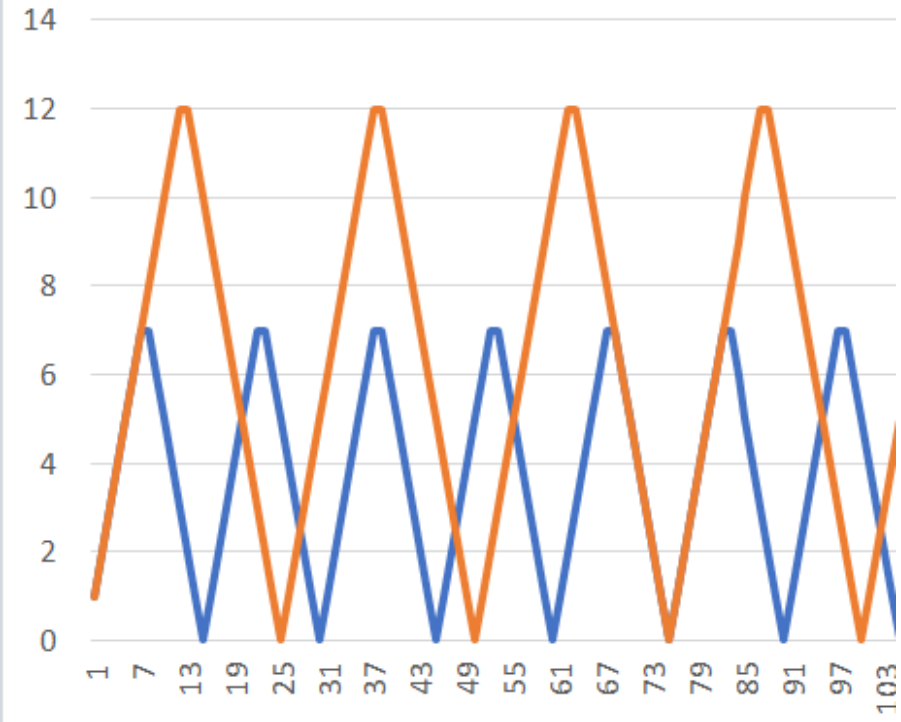
а затем скопировал ячейку B3 во все ячейки третьей строки, начиная с C3.

В ячейки A2 и A3 Петя поместил числа 15 и 25 соответственно.

После этого Петя построил графики для значений во второй и третьей строках, начиная со столбца B, разместив их на одной диаграмме. По оси абсцисс значения совпадают со значениями в первой строке таблицы.

Будем считать областью совпадения графиков такую последовательность не менее чем из двух идущих подряд значений по оси абсцисс, для которой значения во второй и третьей строках таблицы совпадают, то есть графики накладываются друг на друга. Пересечение графиков, то есть единичное значение по оси абсцисс, для которого совпадают значения во второй и третьей строке таблицы, тогда как для предыдущего и последующего значений совпадения нет, не считается областью совпадения графиков.

	A	B	C	D	E	F	G	H	I	J	K
1		1	2	3	4	5	6	7	8	9	10
2	15	1	2	3	4	5	6	7	7	6	5
3	25	1	2	3	4	5	6	7	8	9	10
4											



Анализируя график, Петя обратил внимание, что первая область совпадения графиков соответствует значениям по оси абсцисс от 1 до 7 включительно. Следующая область совпадения графиков соответствует значениям по оси абсцисс от 68 до 82 включительно.

Петя решил поменять значения в ячейках A2 и A3. В ячейку A2 он поместил значение 17, а в ячейку A3 некоторое целое положительное число X. Петя обнаружил, что первая область совпадения графиков по-прежнему находится в начале – в области от 1 до некоторого K по оси абсцисс, а вот вторая область совпадения графиков – от 383 до 399 включительно. Определите и укажите в ответе значение X, которое Петя записал в ячейку A3. Если таких значений несколько, укажите в ответе минимальное из них. Если такого значения не существует, укажите в ответе NULL.

Ответ: 23

3. Сортировка и фильтрация данных, базы данных (2 балла)

[Острова]

Петя разрабатывает компьютерную игру. Игровое поле состоит из 11 островов, обозначенных латинскими буквами от A до K. На острове A стоит замок Героя. Некоторые острова соединены мостами. В рамках одного из заданий Герой должен обойти часть (возможно все) острова и собрать дань с их жителей. При этом, если Герой проходит по мосту, соединяющему два острова, мост рушится и больше по нему пройти нельзя. Задание считается выполненным только, если Герой возвращается на остров со своим замком. Информацию об игровом поле Петя хранит в базе данных, содержащей, в частности, две таблицы: Острова и Мосты. Каждая запись в таблице «Острова» хранит сведения об уникальном идентификаторе записи, названии острова и объеме дани, которую можно собрать с его жителей. Каждая запись таблицы «Мосты» хранит сведения об уникальном идентификаторе записи и двух идентификаторах островов, если между этими островами есть мост.

Петя посмотрел на текущие записи в таблицах:

Острова		
ID	Название	Дань
1	A	0
2	B	20
3	C	10
4	D	20
5	E	70
6	F	10
7	G	20

8	H	30
9	I	30
10	J	50
11	K	30

Мосты		
ID	ID острова 1	ID острова 2
1	1	7
2	1	11
3	2	4
4	3	10
5	4	6
6	4	11
7	5	7
8	7	10
9	8	10
10	9	11

Проанализировав записи, Петя понял, что Герой не сможет выполнить задание. Тогда он добавил Герою возможность в рамках выполнения задания построить один новый мост, связав любые два острова. Определите, какой мост нужно построить, чтобы в результате Герой выполнил задание и при этом собрал максимальную дань. В ответе укажите подряд без пробелов сначала два символа, обозначающие названия островов, которые свяжет новый мост (отсортировав их по возрастанию), а затем число – дань, которую соберет Герой, выполнив задание.

Ответ: BH170

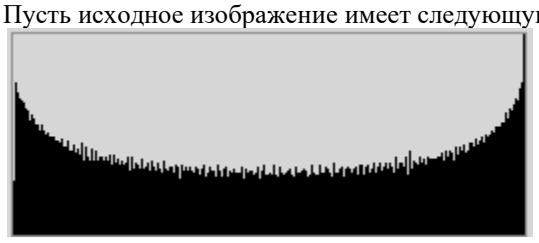
4. Мультимедиа технологии (2 балла)

[Кривые и гистограммы]

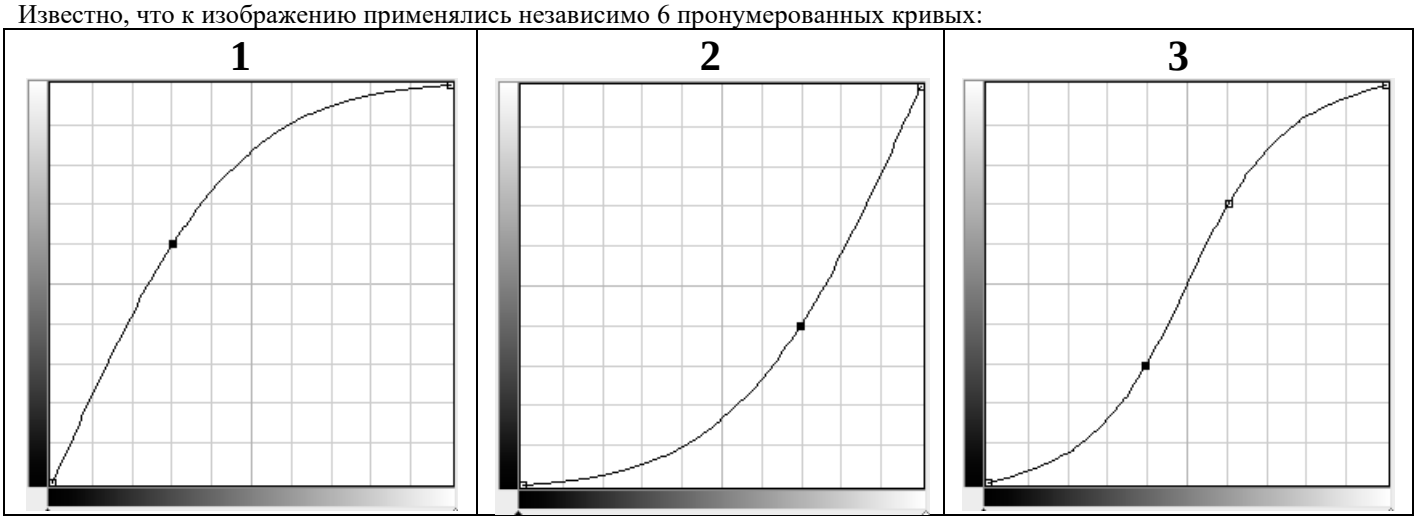
Одним из инструментов оценки распределения яркости пикселей изображения является гистограмма. По оси абсцисс гистограммы расположены возможные значения яркости пикселей слева направо от темных к светлым. По оси ординат – количество пикселей изображения, имеющих такую яркость.

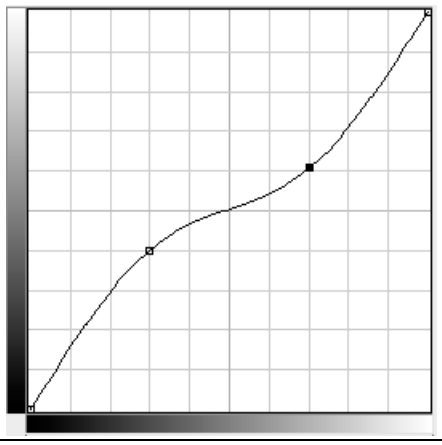
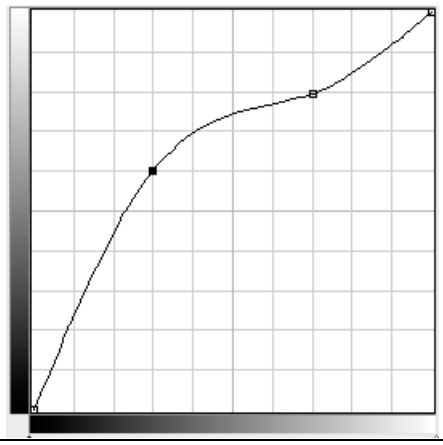
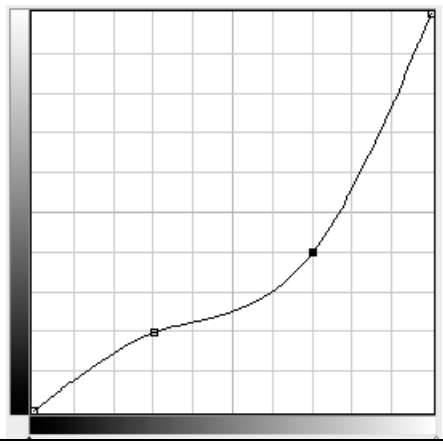
Одним из инструментов яркостной коррекции изображения является Кривая (Curves). Кривая определяет какому значению входной яркости (они отложены по оси абсцисс) соответствует какое значение выходной яркости (они отложены по оси ординат) после коррекции.

Пусть исходное изображение имеет следующую гистограмму:

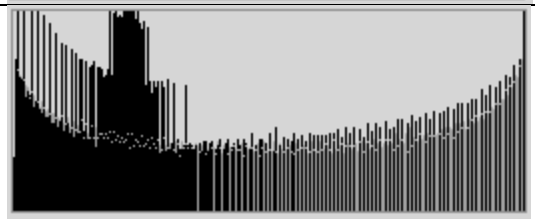
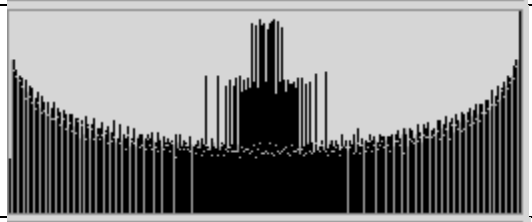
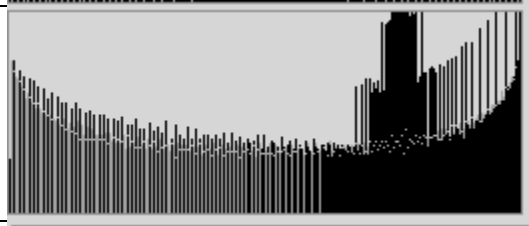
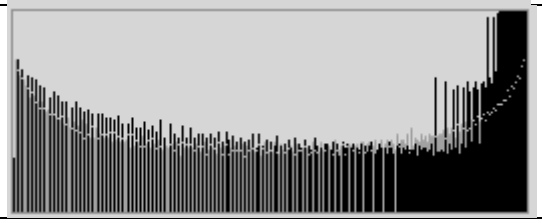


Известно, что к изображению применялись независимо 6 пронумерованных кривых:



4	5	6
		

В результате получились изображения со следующими гистограммами:

1	
2	
3	
4	
5	
6	

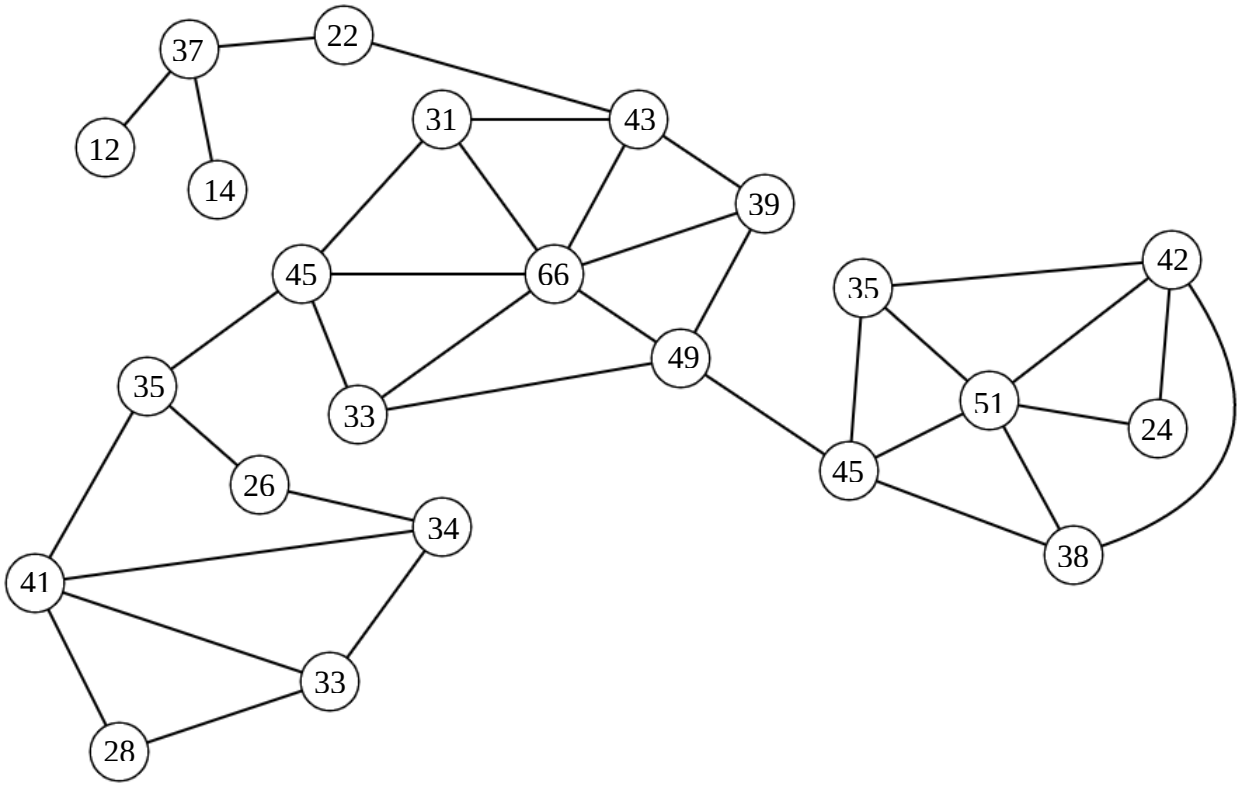
Необходимо определить, какая гистограмма соответствует результату применения какой кривой. В ответе нужно привести последовательность из 6 чисел – номера кривых в порядке возрастания номеров гистограмм. Например, ответ 351246 будет означать, что гистограмма с номером 1 получена в результате применения кривой номер 3, гистограмма с номером 2 была получена в результате применения кривой 5 и т.д.

Ответ: 362451

5. Телекоммуникационные технологии (2 балла)

[Граф сети]

Петя проектирует модель распределенной сетевой инфраструктуры. Инфраструктура может быть представлена в виде графа и состоит из вычислительных узлов, на которых будут размещаться различные сервисы (вершины графа) и сетевых соединений (ребра графа). Граф Пети выглядит следующим образом.



Числа, указанные на узлах – оценка стоимости подключения нового соединения к соответствующему узлу. Стоимость добавления любого нового соединения равна сумме чисел, указанных на соединяемых узлах, независимо от ранее добавленных сетевых подключений. Вася посмотрел проект и сказал, что по требованиям надежности необходимо, чтобы между любыми двумя узлами существовало не меньше двух различных путей, в которых нет одинаковых сетевых соединений. Вася предложил Пете добавить одно или несколько сетевых соединений, чтобы проект стал соответствовать требованиям надежности. Но поскольку Петя экономный, он хочет также сделать это так, чтобы суммарная стоимость добавления всех новых сетевых соединений оказалась минимальной. Определите эту минимальную суммарную стоимость и укажите в ответе как целое число.

Ответ: 76

6. Операционные системы (3 балла)

[Буфер]

Одним из способов взаимодействия процессов в операционной системе является обмен данными через именованный канал. С точки зрения пользователя, он выглядит как файл, в который один процесс записывает данные, а другой считывает, но по факту является буфером в памяти. Как и любой буфер, он ограничен. Представим себе модель из двух процессов P1 и P2 и именованного канала с буфером, размером N символов. Процесс P1 формирует поток символов, которые с помощью именованного канала передает процессу P2.

В модели реализован процессор, который в один момент времени может исполнять только один процесс, при этом процессы образуют циклическую очередь (Round Robin) – им по очереди дается квант непрерывного выполнения – определенное количество условных временных тактов, в которые процесс исполняется, а далее исполняется следующий процесс в очереди и так по кругу. Временем переключения процессора с процесса на процесс пренебрегаем. Никаких других процессов, кроме P1 и P2 в модели нет.

Пусть процесс P1 должен до своего завершения сгенерировать и передать процессу P2 последовательность из 1000 символов. При этом, во время своего исполнения он помещает в буфер именованного канала 11 символов за один условный временной такт. Процесс 2 во время своего исполнения считывает из буфера символы со скоростью 9 символов за один условный временной такт. Квант непрерывного выполнения составляет 3 условных временных такта. Первым возможность вычисления на процессоре получает процесс P1. Если процесс P1 попытается записать символ в заполненный буфер, произойдет авария. При каком минимальном N процесс P1 успешно сможет передать всю последовательность символов без аварии. В ответе укажите целое число.

Ответ: 207

7. Технологии программирования (2 балла)

[Роботы]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод

Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

Настольная игра «Роботы» задается полем размера n×m, в каждой клетке которого могут находиться:

- свободные клетки ('.')
- движущие механизмы, задающиеся символами '<', '^', '>' и 'v';
- стены с шипами ('W');
- ремонтные станции ('R').

Пример поля размера 12×312×3 можно видеть ниже:

```
>>>v..WWW>>>
^<<<.WR....W
..W...>>^<<R
```

Фигурки игроков, которыми можно ходить — роботы. Каждый робот может быть повернут «лицом» в любом из четырех направлений: вверх, вправо, вниз, влево. У i-го робота изначально есть q HP (жизней).

Игроки размещают своих роботов на попарно различные свободные клетки поля в желаемом направлении, после чего происходит k ходов игры. Каждый ход проходит в три последовательные фазы:

- игроки выполняют перемещения;
- игроки одновременно стреляют вперед перед собой на максимальное расстояние до первого попадания в стену, за границу поля или в другого игрока; это означает, что каждый активный робот успевает выстрелить, весь урон наносится моментально, и роботы не снимаются с поля до завершения фазы;
- применяются эффекты поля (движущие механизмы, ремонтная станция).

Во время фазы перемещений игроки выкладывают перед собой d карточек действий, каждая из которых может быть

- поворотом на 90° по или против часовой стрелки («turn right», «turn left»);
- разворотом на 180° («uturn»);
- движением вперед («move forward») или назад («move backward») на одну клетку.

На каждой карточке перемещения указан ее приоритет — уникальное целое число. Во время фазы перемещений сначала все игроки выполняют первое действие в порядке уменьшения его приоритета, затем второе (так же в порядке уменьшения приоритета), и так далее.

Все возможные сценарии взаимодействий в игре, помимо перемещений, описаны ниже:

1. При попытке движения в клетку со стеной игрок не двигается, но теряет 1 HP.
2. При попытке движения в клетку с другим игроком игрок двигается сам и двигает всех игроков перед собой. Если непосредственно за цепочкой игроков находится стена, никто не двигается, но стоящий у стены игрок теряет 1 HP.
3. При попытке движения за границу поля робот теряет все HP, выбывает из игры и снимается с поля.
4. Если в игрока попадает выстрел другого игрока, он теряет 1 HP.
5. Эффекты поля применяются к роботам один раз по очереди: сначала к первому игроку, затем ко второму, и так далее.
6. На фазе применения эффектов поля роботы, находящиеся на движущем механизме, перемещаются на одну клетку в направлении его стрелки. Если в направлении перемещения есть препятствие, см. пункты 1. и 2.
7. На фазе применения эффектов поля роботы, находящиеся на ремонтной станции, восстанавливают 1 HP. Если робот уже имеет q HP, его HP не изменяется.
8. Если в какой-либо момент игры у робота 0 или меньше HP, он выбывает из игры и снимается с поля.
9. Если игрок выбыл из игры, он продолжает выкладывать карточки движений, чтобы сжигать колоду, но эти действия не оказывают никакого влияния на игру.

Вам дано полное описание игры: конфигурация поля, изначальные позиции всех игроков, а также описание всех их действий за k ходов. Определите для каждого игрока его состояние после этих k ходов: если игрок все еще в игре, то его позицию, направление и HP, иначе — место и причину смерти.

Формат входных данных

В первой строке ввода дано одно целое число t — количество тестовых наборов ($1 \leqslant t \leqslant 80$). Далее следуют t тестовых наборов. Перед каждым тестовым набором находится пустая строка. Описание одного тестового набора состоит из описания поля, описания стартового положения игроков и описания ходов.

Первая строка описания поля содержит два целых числа h и w — высоту и ширину поля ($1 \leqslant h,w \leqslant 40$). Следующие h строк состоят из w символов каждая и задают поле. Все возможные символы на поле — из множества символов, перечисленных в начале условия.

Описание стартового положения игроков начинается со строки, содержащей два целых числа n и q — количество игроков и изначальный HP каждого робота ($1 \leqslant n \leqslant 15$; $1 \leqslant q \leqslant 10$). В i-й из следующих строк через пробел даны два целых числа r_i, c_i и символ 'U', 'R', 'D' или 'L' — номер строки и столбца клетки, в которую встает i-й игрок, и его направление (вверх, вправо, вниз или влево, соответственно) ($1 \leqslant r_i \leqslant h$; $1 \leqslant c_i \leqslant w$).

Гарантируется, что все стартовые положения игроков различны и соответствуют свободным клеткам поля.

Первая строка описания ходов содержит два целых числа k и d — количество ходов и количество карточек действий, выкладываемых каждый ход ($1 \leqslant k \leqslant 80$; $1 \leqslant d \leqslant 4$). Далее следуют описания k ходов с первого по k-й. Описание каждого хода состоит из n+1-й строки. Первая из них содержит единственный символ — '$'. Затем i-я из следующих n строк задает действия i-го игрока в формате (<Действие> <Приоритет>), разделенные друг от друга стрелкой « -> ».

Гарантируется, что <Действие> всегда является одной из данных в условии строк, описывающих действия, а <Приоритет> — уникальное (в рамках всей игры) целое неотрицательное число до 10^9.

Формат выходных данных

Для каждого из t тестовых наборов выведите строку «Game <Номер игры>» (где <Номер игры> — номер тестового набора от 1 до t), после чего выведите состояние всех игроков на момент окончания последнего хода.

Описание состояний игроков состоит из n строк, i-я из которых имеет вид (ACTIVE <Строка> <Столбец> <Направление> <HP>), если игрок еще в игре, и (FAILED <Строка> <Столбец> <Причина смерти> <Последний ход>) иначе.

В последнем случае <Строка>, <Столбец> и <Последний ход> задают позицию и номер хода (от 1 до k), на которых игрок выбыл из игры, а <Причина смерти> может быть либо «SHOT», если игрок потерял последний HP от чьего-то выстрела, либо «WALL», если от перемещения в стену, или «BOUNDS», если от перемещения за границу поля (в таком случае позиция выбывания — та клетка, с которой было совершено перемещение за границу).

Пример

Стандартный ввод	Стандартный вывод
2 1 5 <...R 3 10 1 2 R 1 3 U 1 4 L 1 3 $ move forward 2 -> uturn 13 -> move backward 1 turn right 43 -> move backward 3 -> move forward 31 move forward 9 -> move forward 6 -> move forward 12 2 3 ..W W.. 2 3 1 1 R 2 3 L 3 2 $ move forward 19 -> turn right 4 move forward 53 -> turn right 51 $ turn left 2 -> turn right 87 uturn 6 -> uturn 44 $ uturn 1 -> uturn 7 turn right 22 -> turn left 31	Game 1 FAILED 1 1 BOUNDS 1 FAILED 1 1 BOUNDS 1 ACTIVE 1 2 L 9 Game 2 FAILED 1 2 SHOT 3 FAILED 2 2 SHOT 3

Примечание

В первом примере на единственном ходу:

- первым действием сначала второй игрок повернется (теперь он смотрит вправо), затем третий сдвинет всех (включая себя) влево, затем первый сдвинет всех обратно;
- вторым действием сначала первый игрок развернется (и будет смотреть влево), затем третий игрок сдвинет всех влево, после чего второй сделает движение назад и столкнет первого с доски;
- третьим действием второй игрок сдвинется вперед и снова станет соседним с третьим, после чего третий обратно сдвинет его на край доски.

После этого второй и третий роботы выстрелят и попадут друг в друга, а затем движущий механизм сбросит второго робота с доски.

Во втором примере оба робота сдвигаются на позиции (1,2) и (2,2), поворачиваются друг к другу, после чего в конце каждого хода совершают выстрелы друг в друга.

8. Технологии программирования (4 балла) [Видеоконференция]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

Проходит видеоконференция, в которой принимают участие n спикеров. Видео всех участников в конференции отображаются в прямоугольном окне браузера ширины w и высоты h, каждый участник занимает определенный прямоугольный фрагмент экрана со сторонами, параллельными границам экрана.

Разумеется, каждому участнику конференции хочется, чтобы его было видно. При этом требуется соблюдать определенные правила, чтобы картинка на экране выглядела гармонично. А именно,

- фрагменты всех участников должны иметь одинаковую высоту;
- отношение ширины к высоте фрагмента каждого участника должно лежать между 1 и r ($r \geqslant 1$) включительно;
- фрагменты участников не должны пересекаться (накладываться друг на друга), но могут касаться сторонами;
- весь экран должен быть покрыт трансляциями участников, то есть не должно существовать части экрана, не покрытой ни одним фрагментом.

Обратите внимание, что не запрещено делать отношение ширины к высоте разным у разных участников.

Несложно показать, что при выполнении всех условий экран разбивается на горизонтальные полосы одинаковой высоты, которые, в свою очередь, разбиваются на фрагменты трансляций участников. Определите, на сколько полос следует разбить окно браузера и сколько участников разместить в каждой полосе, чтобы все описанные условия могли быть выполнены.

Формат входных данных

В первой строке ввода дано одно целое число t — количество тестовых наборов ($1 \leqslant t \leqslant 10^5$). Далее следуют t тестовых наборов.

Тестовый набор описывается одной строкой, в которой через пробел перечислены три целых числа n, w и h, и вещественное число r — количество участников конференции, размер экрана и ограничение на отношение ширины к высоте фрагмента ($1 \leqslant n \leqslant 10^6$; $1 \leqslant w,h \leqslant 10^7$; $1 \leqslant r \leqslant 2$). Число r дано с точностью 5 знаков после запятой.

Формат выходных данных

Выведите по очереди t ответов на все тестовые наборы.

В качестве ответа выведите на одной строке через пробел: количество полос в разбиении экрана на фрагменты, минимальное число участников в полосе и максимальное число участников в полосе.

Если разместить участников указанным образом невозможно, вместо этого выведите в качестве ответа на соответствующий тестовый набор единственное число −1.

Пример

Стандартный ввод	Стандартный вывод
7 2 4 4 2.00000 5 12 8 1.60000 2 19 5 1.99999 2 20 5 1.99999 9 10 10 1.00000 18 55 40 1.37500 18 55 40 1.37499	2 1 1 2 2 3 1 2 2 -1 3 3 3 4 4 5 -1

Задания для 9 и 10 класса

Заключительный этап, 10 класс (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (3 балла)

[А и В сидели на трубе]

Определите две последние цифры шестнадцатеричной записи числа $0,AB_{16}^N$ при N=20 000 000 000$_{10}$.

В ответе запишите подряд две шестнадцатеричные цифры в порядке их следования в числе.

Ответ: 01

Решение:

Докажем, что $0,AB_{16}^N = \frac{AB_{16}^N}{100_{16}^N}$, например так:

$0,AB_{16}^N = X$

$0,AB_{16}^N * 100_{16}^N = X * 100_{16}^N$

$(0,AB * 100)_{16}^N = X * 100_{16}^N$

$AB_{16}^N = X * 100_{16}^N$

$X = \frac{AB_{16}^N}{100_{16}^N}$

$0,AB_{16}^N = \frac{AB_{16}^N}{100_{16}^N}$

Обратим внимание, что деление на 100_{16}^N приведет просто к изменению позиции запятой и, следовательно, последние две цифры шестнадцатеричной записи числа $0,AB_{16}^N$ будут ровно такими же, как последние две цифры числа AB_{16}^N.

Определение значения AB_{16}^N при N=20 000 000 000 «в лоб» вычислительно затруднено. Но легко доказать, что если построить ряд вида AB_{16}^1, AB_{16}^2, AB_{16}^3, …,

то последние две цифры шестнадцатеричных записей этих чисел будут образовывать период. Действительно, значения последних двух цифр очередного члена ряда будут зависеть только от значений двух последних цифр предыдущего члена ряда, а поскольку количество комбинаций этих цифр конечно, рано или поздно возникнет пара цифр, которые уже ранее встречались, и начнется новый период.

Получить такой период для AB_{16}^N можно, например, программно. Он будет включать в себя 64 значения:

AB, 39, 13, B1, 3B, 69, 23, 61, CB, 99, 33, 11, 5B, C9, 43, C1, EB, F9, 53, 71, 7B, 29, 63, 21, 0B, 59, 73, D1, 9B, 89, 83, 81, 2B, B9, 93, 31, BB, E9, A3, E1, 4B, 19, B3, 91, DB, 49, C3, 41, 6B, 79, D3, F1, FB, A9, E3, A1, 8B, D9, F3, 51, 1B, 09, 03, 01.

Заметим, что следующим значением снова будет AB, что начнет новый период.

Тогда для решения задачи нам нужно вычислить остаток от деления N на 64 (при N=20 000 000 он равен 0) и взять из получившегося набора значений значение с соответствующим номером. Поскольку 20 000 000 делится на 64 нацело, необходимо взять последнее значение. Таким образом, получается ответ «01».

2. Кодирование информации. Объем информации (2 балла)

[Небо, лес и скалы]

Вася хранит фотографию природы формата 3N пикселей на 4N пикселей. Формат хранения предполагает, что изображение может содержать 65536 различных цветов, и для хранения каждого пикселя используется одинаковое для всех пикселей минимально возможное количество бит (хранятся только коды пикселей). Вася разработал алгоритм, позволяющий выделить на фотографиях лес, небо и скалы (Вася считает, что фотографировал только их, и его алгоритм относит каждый пиксель к одной из этих категорий). После чего он выяснил, что в фрагментах, определяемых как небо встречаются всего 2048 различных цветов, в определяемых как лес – 16384 различных цветов, а в определяемых как скалы – в 2 раза меньше, чем в определяемых как небо. Вася изменил формат хранения данных так, чтобы теперь минимально возможное количество пикселей определялось отдельно для леса, отдельно для скал и отдельно для неба (хранятся также только коды пикселей, но пиксели каждого из трех фрагментов кодируются независимо). После изменения формата хранения его фотография стала занимать на 5096 Кбайт меньше. Вася считает, что пикселей, относящихся к фрагментам леса, неба и скал на его фотографии поровну. Определите минимальное N, при котором это возможно.

Ответ: 896

Решение:

Составим уравнение для решения этой задачи. До обработки фото занимает $3N * 4N * \log_2 65536 = 192 * N^2$ бит. После обработки фото будет занимать $N * 4N * \log_2 16384 + N * 4N * \log_2 2048 + N * 4N * \log_2 \left(\frac{2048}{2}\right) = 56N^2 + 44N^2 + 40N^2 = 140N^2$ бит. Разница составляет 5096 Кбайт, т.е. $192N^2 - 140N^2 = 5096 * 1024 * 8$. Отсюда получаем, что $N^2 = \frac{5096*1024*8}{192-140}$.

Таким образом, $N = \sqrt{802816} = 896$.

3. Основы логики (3 балла)

[Кто в системе?]

Дана система логических уравнений. Определите, сколько существует различных наборов переменных A, B, C и D, при которых выполняется условие истинности системы.

$$\begin{cases} ((A \wedge B) XOR (B \wedge C)) XOR (C \wedge D) = 1 \\ ((A \vee B) \rightarrow (B \vee \mathrm{C})) \rightarrow (\mathrm{C} \vee \mathrm{D}) = 1 \\ ((A \vee B) \mathrm{XOR} (B \vee \mathrm{C})) \mathrm{XOR} (\mathrm{C} \vee \mathrm{D}) = 1 \\ ((A \vee B) \leftarrow (B \vee \mathrm{C})) \leftarrow (\mathrm{C} \vee \mathrm{D}) = 1 \end{cases}$$

Примечание: ∧ - операция коньюнкции, ∨ - логическая дизъюнкция, → - импликация, ← - обратная импликация, XOR – исключающее или, 1 – истина. Операция обратной импликации эквивалентна операции $(A \vee \bar{B})$.

Ответ: 4

Решение:

Составим таблицу истинности всех выражений. Очевидно, что система обращается в 1, только если все выражения обращаются в 1.

A	B	C	D	((A*B)^(B*C)^(C*D))	((A+B)->(B+C)->(C+D))	((A+B)^(B+C)^(C+D))	((A+B)<-(B+C)<-(C+D))
0	0	0	0	FALSE	FALSE	FALSE	TRUE
0	0	0	1	FALSE	TRUE	TRUE	TRUE
0	0	1	0	FALSE	TRUE	FALSE	FALSE
0	0	1	1	TRUE	TRUE	FALSE	FALSE
0	1	0	0	FALSE	FALSE	FALSE	TRUE
0	1	0	1	FALSE	TRUE	TRUE	TRUE
0	1	1	0	TRUE	TRUE	TRUE	TRUE
0	1	1	1	FALSE	TRUE	TRUE	TRUE
1	0	0	0	FALSE	TRUE	TRUE	TRUE
1	0	0	1	FALSE	TRUE	FALSE	TRUE
1	0	1	0	FALSE	TRUE	TRUE	TRUE
1	0	1	1	TRUE	TRUE	TRUE	TRUE
1	1	0	0	TRUE	FALSE	FALSE	TRUE
1	1	0	1	TRUE	TRUE	TRUE	TRUE
1	1	1	0	FALSE	TRUE	TRUE	TRUE
1	1	1	1	TRUE	TRUE	TRUE	TRUE

Из таблицы видно, что все выражения обращаются в 1 в 4 случаях – при 0110, 1011, 1101 и 1111.

Также возможно программное решение задания, например, такое:

```
from itertools import product

def f1(a, b, c, d):
    return ((a and b) != (b and c)) != (c and d)

def f2(a, b, c, d):
    return ((a or b) <= (b or c)) <= (c or d)

def f3(a, b, c, d):
    return ((a or b) != (b or c)) != (c or d)

def f4(a, b, c, d):
    return ((a or b) >= (b or c)) >= (c or d)

res = 0
for x in product((False, True), repeat=4):
    res += f1(*x) and f2(*x) and f3(*x) and f4(*x)
print(res)
```

4. Кодирование информации. Алгоритмы обработки кодированной информации (2 балла) [Очень длинная строка]

Последовательности символов строятся по следующим правилам:

Исходная последовательность состоит из двух букв: “AA”. Далее на каждом шаге берется очередная буква английского алфавита, и пара таких букв добавляется в начало последовательности и после каждой буквы последовательности, полученной на предыдущем шаге. Например, последовательности, получаемые после первых 3 шагов, будут выглядеть таким образом (третья последовательность разбита на две строки для лучшего восприятия):

4. BBABBABB
5. CCBCCBCCACCBCCBCCACCBCCBCC
6. DDCDDCDDBDDCDDCDDBDDCDDCDDADDCDDCDDBDDCDDCDDBDDCDDCDDADDCDDCDDBDDCDDCD DBDDCDDCDD

Построение завершается, когда в результате очередного шага в последовательности впервые появляются буквы Z. Эта последовательность является результирующей.

Определите, какие буквы стоят в результирующей последовательности на позициях 961 376 769 и 1 035 574 967 097, считая от начала строки с 1. В ответе укажите их подряд в указанном порядке.

Ответ: КС

Решение:

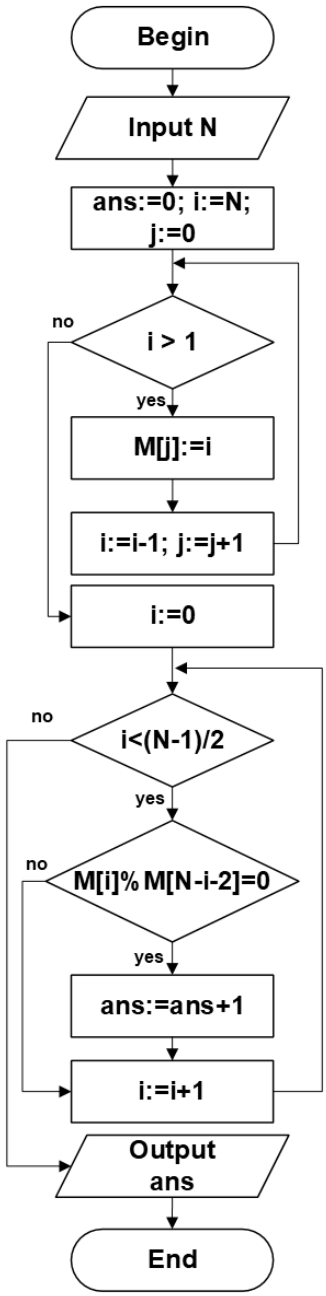
Обратим внимание, что в строке, в которой впервые появляется очередная буква алфавита, эта буква стоит на всех позициях, кроме тех, номер которых кратен 3 (считая с 1 от начала строки). А на позициях, кратных 3 стоят буквы, которые встречались в предыдущих строках. Также заметим, что при переходе к очередной строке позиция каждой буквы из предыдущей строки увеличивается в 3 раза. Так, буква B, которая встретилась впервые на второй позиции в строке «1», в строке «2» будет на шестой позиции, в строке «3» - на восемнадцатой позиции и т.д.

Строка с буквой Z будет 25-ой по счету. Рассмотрим число 961 376 769. Легко убедиться, что оно кратно 3, следовательно, это не буква Z. Если мы его поделим на 3, то получим 320 458 923 – это число также кратно 3, следовательно, в строке, в которой впервые появился символ Y, искомая буква стоит на позиции, кратной 3, то есть это не Y. Представим анализируемое число, как $k*3^n$, где k – сомножитель, не кратный 3. Получится, что 961 376 769 = $67*3^{15}$. Отсчитаем от Y в обратном порядке 15-ую букву – это буква «К». Мы определили первую часть ответа. Аналогично, 1 035 574 967 097 = $11*3^{23}$. Отсчитаем в обратном порядке от Y 23-ю букву и получим вторую часть ответа – «С».

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (2 балла)

[Считаем ноли]

Дана блок-схема алгоритма. M – массив размера N – 1. Операция A%B означает взятие остатка от деления A на B.



Алгоритм последовательно запустили со всеми натуральными значениями N от 100 до 199 включительно. Определите, при каком значении N был выведен наименьший результат (назовем его величиной A), а при каком – наибольший (назовем его величиной B). В случае, если величина A выводилась несколько раз – определите наименьшее из значений N, при которых это произошло. В случае, если величина B выводилась несколько раз – определите наибольшее из значений N, при которых это произошло. В ответе укажите через пробел 2 числа – сначала значение N для величины A и затем значение N для величины B.

Примечание: A/B означает операцию целочисленного деления A на B.

Ответ: 101 178

Решение:

Проэмулируем описанные в задаче действия, реализовав алгоритм на языке C++. Обратите внимания, для корректного определения N, при котором выводилась величина A, необходимо не обновлять значение в случае, если величина A встретилась повторно. В случае с величиной B – наоборот, каждый раз, когда выводимая величина равняется B, ответ необходимо обновить.

```
#include <iostream>
#include <vector>

using namespace std;
vector<int> numbers; //динамический массив чисел от N до 2

int main()
{
  int a = INT_MAX, b = 0, a_n = 0, b_n = 0;
  for (int n = 100; n < 200; n++)
  {
    int ans = 0; //счетчик количества 0
    numbers.clear();
    for (int i = n; i > 1; i--) //заполним динамический массив числами от текущего n до 2
    {
      numbers.push_back(i);
    }
    for (int i = 0; i < (n - 1) / 2; i++) //вычислим остатки
    {
      if (numbers[i] % numbers[n - i - 2]==0) ans++; //подчет числа 0
    }
    cout << ans << endl;
    if (ans < a) //определяем, нужно ли обновить величину A и ответ для нее
    {
      a = ans;
      a_n = n;
    }
    if (ans >= b) //определяем, нужно ли обновить величину B и ответ для нее
    {
      b = ans;
      b_n = n;
    }
  }
    cout << a_n << " " << b_n;
```

Пример реализации на Python:

```
max_ = -10 ** 9
min_ = 10 ** 9
max_n = min_n = 100
for n in range(100, 200):
    ans = 0
    m = [0] * n
    j = 0
    for i in range(n, 1, -1):
        m[j] = i
        j += 1
    for i in range((n - 1) // 2):
        if m[i] % m[n - i - 2] == 0:
            ans += 1
    if ans < min_:
        min_ = ans
        min_n = n
    if ans >= max_:
```

```
        max_ = ans
        max_n = n
print(min_n, max_n)
```

6. Телекоммуникационные технологии (1 балл) [Адреса и маски]

Маска сети для IPv4 адресации – это 4-х байтное число, которое делит IP адрес на адрес сети (первая часть) и адрес узла (вторая часть). У всех адресов одной IP-сети совпадают первые части и отличаются вторые. Для части IP адреса, соответствующей адресу сети, в маске сети содержатся двоичные единицы, а для части IP адреса, соответствующей адресу узла, в маске сети содержатся двоичные нули. Для записи масок сетей часто используется нотация, когда после IP-адреса через «/» указывается число бит, отводимых в маске под адрес сети. Например, для адреса 11.12.0.8 и маски 255.0.0.0 запись будет иметь следующий вид 11.12.0.8/8. Служебным адресом сети называют адрес, у которого все биты адреса узла (второй части) равны 0. Широковещательным адресом сети называют адрес, у которого все биты адреса узла (второй части) равны 1.

Известно, что широковещательный адрес некоторой сети – 192.121.87.255, а адрес 192.121.83.178 не принадлежит этой сети. Определите служебный адрес этой сети и ее маску. Если существует несколько возможных масок, укажите ту, у которой максимально количество узлов в сети.

Введите ответ на поставленную задачу в виде: A.B.C.D/M, где A, B, C, D – байты служебного адреса сети в десятичной системе счисления, а M – число бит, отводимых под адрес сети.

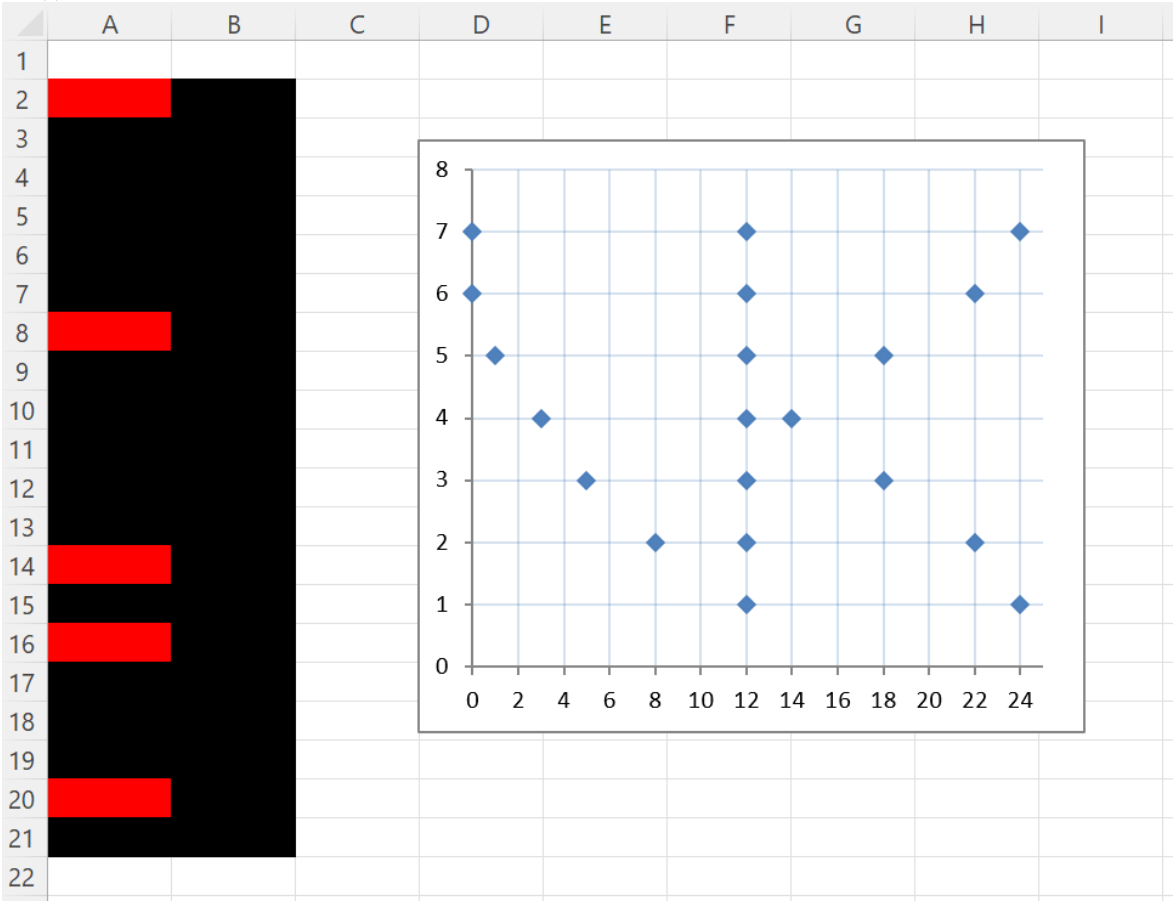
Ответ: 192.121.84.0/22

Решение:

Представим широковещательный адрес сети в двоичном формате: 192.121.87.255 = 11000000.01111001.01010111.11111111. По определению широковещательного адреса, первые 21 бит маски точно будут относиться к адресу сети, а не адресу узла. Теперь представим в двоичном формате второй известный нам адрес: 192.121.83.178 = 11000000.01111001.01010011. 10110010. Заметим, что первые 21 бит совпадают с широковещательным адресом, а 22-ой различается. Если бы к адресу сети относились первые 21 бит, то второй адрес принадлежал бы сети, а он не принадлежит. Значит, к адресу узла относится больше 21 бита. При этом, количество узлов будет максимально у той подсети, у которой минимальное количество бит относится к адресу сети. Минимальное число бит в маске больше 21 – это 22. Значит, наша искомая сеть имеет в двоичном формате вид 11000000.01111001.01010100.00000000/22, что в десятичном представлении записывается как 192.121.84.0/22.

7. Технологии обработки информации в электронных таблицах. Технологии сортировки и фильтрации данных (1 балл) [VK]

Петя изучает различные виды диаграмм в электронных таблицах. Для того, чтобы потренироваться в построении точечной диаграммы с маркерами, Петя решил изобразить логотип IT-компании VK, заполнив целыми неотрицательными числами диапазон A2:B21:



Известно, что числа в столбце B отсортированы по невозрастанию. Какая минимальная сумма может быть у чисел в красных ячейках. В ответе укажите целое число.

Примечание. При построении точечной диаграммы с маркерами значения из левого столбца используются в качестве координат маркеров по оси абсцисс, а значения из правого столбца - для координат маркеров по оси ординат.

Ответ: 30

Решение:

Заметим, что на диаграмме ровно 20 маркеров, следовательно, каждому маркеру соответствует ровно одна пара значений в диапазоне A2:B21. Поскольку значения в столбце B (правом столбце) соответствуют значениям координат точек по оси ординат, легко построить значения этого столбца в указанном порядке сортировки. Сразу отметим позиции красных ячеек в соответствии с условием:

	7
	7
	7
	6
	6
	6
	5
	5
	5
	4
	4
	4
	3
	3
	3
	2
	2
	2
	1
	1

Следовательно, нам нужно одно минимальное значения для маркера с ординатой 7, одно минимальное значение для маркера с ординатой 5, два минимальных значения для маркера с ординатой 3 и одно минимальное значение для маркера с ординатой 1. Из графика можно получить, что это числа 0, 1, 5, 12, 12. Их сумма будет равна 30.

8. Технологии программирования (2 балла) [Тестирование мессенджера]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Перед выпуском VK Messenger'a разработчики из IT-компании «VK», как и положено, убеждаются в корректности работы приложения. Проверкой корректности работы систем занимаются тестировщики и QA-инженеры.

Часть функциональных тестов для тестирования смены ников выглядит следующим образом:

- генерируется случайный сценарий взаимодействия пользователей с приложением;
- в приложении симулируется выполнение этого сценария;
- проверяется корректность итогового состояния приложения.

Вам предлагается почувствовать себя в роли тестировщика и реализовать решение, обрабатывающее сценарий без взаимодействия с приложением. Результат обработки сценария затем будет сравниваться с итоговым состоянием приложения.

Каждый сценарий состоит из трех наборов событий.

Первый набор состоит из событий вида «в момент времени t_i поступил запрос регистрации нового пользователя с ID id_i и ником $handle_i$». Гарантируется, что ID уникальны для всех пользователей приложения. Если выбранный ник никем не занят, регистрация проходит успешно, иначе запрос на регистрацию отклоняется. Во втором случае пользователь с тем же ID может попробовать зарегистрироваться еще раз.

Второй набор описывает события вида «в момент времени t_i пользователь с ником $handle_{i,1}$ отправляет запрос на смену ника на $handle_{i,2}$». Гарантируется, что для каждого такого запроса ник $handle_{i,1}$ кому-то принадлежит. Если $handle_{i,2}$ уже занят каким-либо пользователем, запрос отклоняется, иначе пользователь успешно меняет ник. При успешной смене ника старый ник перестает ассоциироваться с каким-либо пользователем, пока кто-то снова его не займет.

Третий набор описывает события вида «в момент времени t_i пользователь с ником $handle_{i,1}$ отправляет сообщение пользователю с ником $handle_{i,2}$». Гарантируется, что и $handle_{i,1}$, и $handle_{i,2}$ на момент времени t_i соответствуют каким-то зарегистрированным пользователям.

Также гарантируется, что никакие два события не происходят в одно и то же время, то есть все t_i уникальны.

Вам даны описания всех трех наборов событий. Определите для каждого пользователя, сколько он всего отправил сообщений, и какому пользователю он отправил больше всего сообщений за весь сценарий.

Формат входных данных

В первой строке ввода дано единственное целое число T — количество сценариев, которое вам необходимо обработать ($1 \le T \le 100$).

Далее следуют T описаний сценариев. Описание каждого сценария начинается с пустой строки, после чего следуют три набора событий. В первой строке описания q-го набора (q от 1 до 3) дано единственное целое число n_q — количество событий в наборе, после чего следуют n_q строк в указанном ниже формате ($1 \le n_1+n_2+n_3 \le 1000;\ 0 \le n_q$).

- События первого набора задаются в формате «t_i: REG id_i $handle_i$».
- События второго набора задаются в формате «t_i: CHANGE $handle_{i,1}$ $handle_{i,2}$».
- События третьего набора задаются в формате «t_i: SEND $handle_{i,1}$ $handle_{i,2}$».

Моменты событий t_i — целые числа от 1 до 10_9. Также все id_i — целые числа от 1 до 10_9, а $handle_i$ — строки из маленьких латинских букв длины не более 10.

Гарантируется, что в каждом наборе входных данных все t_i различны. Гарантируется, что успешно зарегистрированный пользователь не предпринимает попытки зарегистрироваться еще раз.

Формат выходных данных

Для каждого сценария выведите требуемую статистику для каждого пользователя.

В первой строке статистики выведите целое число q — количество зарегистрированных пользователей. В следующих q строках выведите статистику для каждого пользователя в порядке возрастания их ID в формате «<id> SENT <total> TOP <topcount$_{top}$> TO <id$_{top}$>», где total — суммарное количество отправленных пользователем сообщений, id_{top} — ID пользователя, получившего от него больше всего сообщений, а $count_{top}$ — само количество сообщений, отправленных пользователю id_{top}.

Если у некоторого пользователя есть несколько собеседников, получивших от него максимальное число сообщений, выведите в качестве id_{top} минимальный из их ID. Если пользователь не отправлял сообщения, считайте $count_{top}$ равным 0.

Пример

Стандартный ввод	Стандартный вывод
2 4 10: REG 1 sh 11: REG 2 sh 12: REG 3 qwerty 15: REG 2 shvetz 2 13: CHANGE sh shvetz 18: CHANGE shvetz sh 4 14: SEND shvetz qwerty 16: SEND shvetz shvetz 17: SEND shvetz qwerty 19: SEND qwerty sh 2 10: REG 1 a 11: REG 2 b 4 14: CHANGE a c 17: CHANGE b a 20: CHANGE c a 23: CHANGE c b 10 12: SEND a b 13: SEND b a 15: SEND b c 16: SEND b c 18: SEND a c 19: SEND a c 21: SEND c a 22: SEND a c 24: SEND a b 25: SEND a b	2 1 SENT 3 TOP 2 TO 3 3 SENT 1 TOP 1 TO 1 2 1 SENT 2 TOP 2 TO 2 2 SENT 8 TOP 8 TO 1

Решение

Задача на реализацию, требовалось сделать ровно то, что требуется по условию. Чтобы обработать события в правильном порядке, надо было выделить момент времени для каждого события, после чего отсортировать события по времени.

Выделить время в первом варианте можно было, удалив из первого слова в строке последний символ, а во втором — прочитав слово после «AT», после чего преобразовав полученное слово в integer.

Затем, после сортировки событий, необходимо их обработать. Будем хранить необходимую статистику по пользователям, а также информацию о том, какие ники и кем сейчас заняты. Для пользователя надо знать общее число отправленных сообщений total[id] и число отправленных сообщений в диалогах с каждым другим пользователем count[id1][id2]. Также, для каждого занятого на данный момент ника handle надо знать ID пользователя, который его занимает, users[handle].

Такую информацию можно хранить в переменных типов dict[int, int], dict[int, dict[int, int]] и dict[str, int] (Python). В Java и C++ ту же роль выполняют HashMap и unoredered_map.

Тогда обработка операций выглядит следующим образом (ниже приведен похожий на Python псевдокод, в реальном решении могут быть отличия из-за специфики выбранного языка):

```
1. регистрация: проверить, что ник не занят, и выдать его новому пользователю
id, handle <- событие
if handle not in users:
    users[handle] = id
    total[id] = 0
2. смена ника: проверить, что новый ник не занят, и поменять его
handle1, handle2 <- событие
id = users[handle1]
if handle2 not in users:
    users[handle2] = id
    users.pop(handle1)
3. сообщение: увеличить соответствующие счетчики в статистике
handle1, handle2 <- событие
id1, id2 = users[handle1], users[handle2]
total[id1]++
count[id1][id2]++
```

В самом конце надо для каждого пользователя вывести total[id] и самого частого собеседника по отправленным/полученным сообщениям. Для этого выпишем все пары «ключ-значение» из count[id] и отсортируем по значению, а при равном значении — по ключу. Требовалось также не забыть аккуратно обработать случай, когда пользователь не отправлял/не получал сообщения, в таком случае для него idtop равен min(users.values()).

9. Технологии программирования (4 балла) [Устранение массива]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Дан массив a длины n. Каждый элемент массива — 0, 1 или 2. Известно, что нули находятся только строго в левой половине массива, то есть на индексах от 1 до ⌊n/2⌋. Все оставшиеся элементы могут быть равны только 1 или 2.

С массивом разрешается выполнять следующие два вида действий:

Заплатить две монеты и удалить из массива любое число a_i. После такого действия массив $[a_1,a_2,\ldots,a_k]$ превращается в $[a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_k]$.

Заплатить одну монету и заменить два стоящих рядом числа a_i и a_{i+1} на a копий числа a_{i+1}. То есть, возможны следующие преобразования:

$$[a_1,\ldots,a_{i-1},0,a_{i+1},\ldots,a_k] \to [a_1,\ldots,a_{i-1},a_{i+2},\ldots,a_k]$$
$$[a_1,\ldots,a_{i-1},1,a_{i+1},\ldots,a_k] \to [a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_k]$$
$$[a_1,\ldots,a_{i-1},2,a_{i+1},\ldots,a_k] \to [a_1,\ldots,a_{i-1},a_{i+1},a_{i+1},\ldots,a_k]$$

Иными словами, 0 можно удалить вместе со следующим числом, 1 можно удалить саму по себе, если она стоит не в конце массива, а 2 можно заменить на следующее за ней число.

Определите, какое минимальное число монет требуется заплатить, чтобы превратить данный массив в пустой. Обратите внимание, что ответ всегда существует, потому что можно просто n раз применить первую операцию.

Формат входных данных

В первой строке дано единственное целое число T — количество наборов входных данных ($1 \le T \le 100$). Далее следуют T наборов входных данных.

Каждый набор входных данных начинается со строки, содержащей единственное число n — изначальную длину массива. В следующей строке через пробел перечислены целые числа a_1, ……, a_n — элементы массива ($0 \le a_i \le 2$).

Гарантируется, что сумма длин массивов по всем наборам входных данных не превосходит 10^6.

Формат выходных данных

Выведите T целых чисел, каждое в своей строке — ответы на задачу для каждого набора входных данных в том порядке, в котором они даны во вводе.

Пример

Стандартный ввод	Стандартный вывод
4 7 0 2 0 1 1 2 2 5 1 1 1 1 1 11 0 1 2 2 0 1 2 2 1 2 1 10 1 2 1 2 0 2 2 2 2 1	5 6 9 11

Решение

Задача решалась некоторым подобием жадного алгоритма. Ниже мы приведем полное решение с доказательством, однако, чтобы придумать решение, не обязательно было его доказывать. Некоторые вещи, например, что (0, 2) → ∅ — самая «выгодная» операция, кажутся интуитивно понятными и без формального обоснования.

Решение задачи выглядит следующим образом.

1. Найдем самый правый 0 в массиве, после чего для всех элементов слева от него по очереди выполним замены (1, 0) → 0 или (2, 0) → (0, 0). Все единицы слева исчезли, все двойки превратились в нули.

2. Пусть у нас получилось ровно p нулей, а справа от них стоят q единиц и r двоек.

$$a = [\overbrace{0, 0, \ldots, 0}^{p}, \overbrace{1, 2, 2, 1, \ldots}^{q+r}]$$

Заметим, что

• от 1 на конце массива и от 2 в любом месте не избавиться никак, кроме как удалив их за две монеты или удалив их операцией (0, x) → ∅;

• от остальных единиц можно избавиться с помощью (1, x) → x;

• операция (2, 1) → (1, 1) «не выгодна», после придется избавляться от получившейся лишней единицы хотя бы за одну монету, тогда как можно было бы просто удалить исходную двойку за две монеты;

3. Если нули располагались только в левой половине массива, p 6 q+r. Поскольку операций вида (0, x) → ∅ может быть не больше, чем количество нулей, то есть p, справа могуть остаться «лишние» элементы, на которых нулей не хватит. Будем удалять «лишние» единицы с помощью операции (1, x) → x, пока это возможно.

4. После этого либо останется q + r = p, и можно будет за p монет удалить все с помощью нулей, либо будет все еще выполняться q + r > p, но справа не останется удаляемых за одну монету единиц.

5. В таком случае либо справа остались только двойки, либо двойки и одна единица на конце. Удалим p двоек с помощью (0, 2) → ∅, а на каждое из оставшихся чисел потратим по две монеты. Если на хвосте оставалась единица, то замены (2, 1) → (1, 1) → 1 тратят столько же монет, сколько и просто удаление одного числа.

Чтобы получить полный балл за задачу, требовалось посчитать p, q и r за линейное время, после чего посчитать ответ формулами, а не выполнять операции в явном виде — удаление из середины массива занимает много времени.

Дальше будет приведено строгое доказательство корректности этого решения. Посмотрим на следующую функцию:

Ф(a) = a.size() + sum(a) + 2C,

где C — количество потраченных монет для получения массива a из исходного. В конце и длина, и сумма элементов массива будут равны нулю, то есть будет выполняться Ф = 2C. Если требуется минимизировать C, то это то же самое, что и минимизировать Ф.

Заметим, что многие из разрешенных операций не меняют значение этой функции. А именно, замены (0, 0) → ∅, (1, x) → x и (2, 0) → (0, 0): C увеличивается на 1, а a.size()+sum(a) уменьшается на 2. Помимо этого, только две операции уменьшают Ф: (0, 1) → ∅ на 1 и (0, 2) → ∅ на 2, а остальные — увеличивают.

Посмотрим на ситуацию (p, q, r) после первого шага решения. Все операции в первом шаге — «бесплатные», они не меняют Ф. Заметим, что уменьшить Ф можно не больше, чем p раз, с помощью каждого из нулей в левой половине. При этом уменьшить Ф на два можно не больше, чем r раз, с помощью (0, 2) → ∅ с каждой двойкой в правой половине.

Если p 6 r, то лучшего результата добиться не получится, потому что каждая двойка справа от последнего нуля удаляется только с помощью (0, 2), либо за две монеты, то есть для минимизации Ф требуется максимизировать количество операций (0, 2) → ∅, что в данном случае сводится к получению как можно большего p, что мы и сделали. Оставшиеся шаги в решении — «бесплатное» удаление лишних единиц и вынужденное удаление всего оставшегося за две монеты.

Если p > r, то может показаться, что могло бы быть выгодно вместо каких-то операций (0, 1) → ∅, уменьшающих Ф на один, сделать больше операций (0, 2) → ∅ в левой части массива, получив p поменьше. Однако если сделать такую операцию в левой части массива, Ф уменьшается на 2, но и p уменьшается на 2, то есть теряются две потенциальные операции с ΔФ > 1, которые в совокупности не хуже.

Таким образом, показанный алгоритм действительно минимизирует количество потраченных монет

Заключительный этап, 9 класс (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла) [Не бином Ньютона]

Дано выражение в системе счисления с основанием 2022.

$$X_{2022} = 11^4_{2022} + 11^3_{2022} + 11^2_{2022} + 11_{2022} + 1_{2022}$$

Определите сумму цифр записи числа X в 2022-ричной системе счисления и представьте результат в десятичной системе счисления. В ответ запишите одно число в десятичной системе счисления.

Ответ: 31

Решение:

Заметим, что вычисление X_{2022} можно упростить, используя формулу сокращенного умножения: $(a^n - b^n) = (a-b)(a^{n-1} + a^{n-2}b + \cdots + ab^{n-2} + b^{n-1})$. В данном случае $a = 11_{2022}, b = 1, n = 5$. Таким образом, искомое выражение можно вычислить как $\frac{a^n - b^n}{a-b}$. $X_{2022} = \frac{11^5_{2022} - 1_{2022}}{11_{2022} - 1_{2022}} = \frac{15AA51_{2022} - 1_{2022}}{10_{2022}} = 15AA5_{2022}$. Вычислим сумму этих цифр в десятичной системе счисления: $1_{2022} + 5_{2022} + A_{2022} + A_{2022} + 5_{2022} = 1_{10} + 5_{10} + 10_{10} + 10_{10} + 5_{2022} = 31_{10}$.

2. Кодирование информации. Объем информации (2 балла) [Небо, лес и скалы]

Вася хранит фотографию природы формата 3N пикселей на 4N пикселей. Формат хранения предполагает, что изображение может содержать 65536 различных цветов, и для хранения каждого пикселя используется одинаковое для всех пикселей минимально возможное количество бит (хранятся только коды пикселей). Вася разработал алгоритм, позволяющий выделить на фотографиях лес, небо и скалы (Вася считает, что фотографировал только их, и его алгоритм относит каждый пиксель к одной из этих категорий). После чего он выяснил, что в фрагментах, определяемых как небо встречаются всего 2048 различных цветов, в определяемых как лес – 16384 различных цветов, а в определяемых как скалы – в 2 раза меньше, чем в определяемых как небо. Вася изменил формат хранения данных так, чтобы теперь минимально возможное количество пикселей определялось отдельно для леса, отдельно для скал и отдельно для неба (хранятся также только коды пикселей, но пиксели каждого из трех фрагментов кодируются независимо). После изменения формата хранения его фотография стала занимать на 5096 Кбайт меньше. Вася считает, что пикселей, относящихся к фрагментам леса, неба и скал на его фотографии поровну. Определите минимальное N, при котором это возможно.

Ответ: 896

Решение:

Составим уравнение для решения этой задачи. До обработки фото занимает $3N * 4N * \log_2 65536 = 192 * N^2$ бит. После обработки фото будет занимать $N * 4N * \log_2 16384 + N * 4N * \log_2 2048 + N * 4N * \log_2 \left(\frac{2048}{2}\right) = 56N^2 + 44N^2 + 40N^2 = 140N^2$ бит. Разница составляет 5096 Кбайт, т.е. $192N^2 - 140N^2 = 5096 * 1024 * 8$. Отсюда получаем, что $N^2 = \frac{5096*1024*8}{192-140}$.

Таким образом, $N = \sqrt{802816} = 896$.

3. Основы логики (3 балла) [Кто в системе?]

Дана система логических уравнений. Определите, сколько существует различных наборов переменных A, B, C и D, при которых выполняется условие истинности системы.

$$\begin{cases} ((A \wedge B) XOR (B \wedge C)) XOR (C \wedge D) = 1 \\ ((A \vee B) \rightarrow (B \vee \mathrm{C})) \rightarrow (\mathrm{C} \vee \mathrm{D}) = 1 \\ ((A \vee B) \mathrm{XOR} (B \vee \mathrm{C})) \mathrm{XOR} (\mathrm{C} \vee \mathrm{D}) = 1 \\ ((A \vee B) \leftarrow (B \vee \mathrm{C})) \leftarrow (\mathrm{C} \vee \mathrm{D}) = 1 \end{cases}$$

Примечание: $\wedge$ *- операция коньюнкции,* $\vee$ *- логическая дизъюнкция,* $\rightarrow$ *- импликация,* $\leftarrow$ *- обратная импликация, XOR – исключающее или, 1 – истина. Операция обратной импликации эквивалентна операции* $(A \vee \bar{B})$.

Ответ: 4

Решение:

Составим таблицу истинности всех выражений. Очевидно, что система обращается в 1, только если все выражения обращаются в 1.

A	B	C	D	((A*B)^(B*C)^(C*D))	((A+B)->(B+C)->(C+D))	((A+B)^(B+C)^(C+D))	((A+B)<-(B+C)<-(C+D))
0	0	0	0	FALSE	FALSE	FALSE	TRUE
0	0	0	1	FALSE	TRUE	TRUE	TRUE
0	0	1	0	FALSE	TRUE	FALSE	FALSE
0	0	1	1	TRUE	TRUE	FALSE	FALSE
0	1	0	0	FALSE	FALSE	FALSE	TRUE
0	1	0	1	FALSE	TRUE	TRUE	TRUE
0	1	1	0	TRUE	TRUE	TRUE	TRUE
0	1	1	1	FALSE	TRUE	TRUE	TRUE
1	0	0	0	FALSE	TRUE	TRUE	TRUE
1	0	0	1	FALSE	TRUE	FALSE	TRUE
1	0	1	0	FALSE	TRUE	TRUE	TRUE
1	0	1	1	TRUE	TRUE	TRUE	TRUE
1	1	0	0	TRUE	FALSE	FALSE	TRUE
1	1	0	1	TRUE	TRUE	TRUE	TRUE
1	1	1	0	FALSE	TRUE	TRUE	TRUE
1	1	1	1	TRUE	TRUE	TRUE	TRUE

Из таблицы видно, что все выражения обращаются в 1 в 4 случаях – при 0110, 1011, 1101 и 1111.

Также возможно программное решение задания, например, такое:

```
from itertools import product

def f1(a, b, c, d):
    return ((a and b) != (b and c)) != (c and d)

def f2(a, b, c, d):
    return ((a or b) <= (b or c)) <= (c or d)

def f3(a, b, c, d):
    return ((a or b) != (b or c)) != (c or d)

def f4(a, b, c, d):
    return ((a or b) >= (b or c)) >= (c or d)

res = 0
for x in product((False, True), repeat=4):
    res += f1(*x) and f2(*x) and f3(*x) and f4(*x)
print(res)
```

4. Алгоритмизация и программирование. Формальные исполнители (1 балл) [Цезарь со сдвигом]

Андрей придумал алгоритм для кодирования строк, составленных из строчных английских букв, на основе шифра Цезаря. Чтобы зашифровать слово, он предпринимает следующие шаги:

1. Пронумеровать буквы английского алфавита от 1 до 26 (по порядку, без пропусков, буквы идут в алфавитном порядке).
2. Определить начальный размер сдвига – N.
3. Определить номер в алфавите каждой буквы исходного слова. Обозначим номер i-ой буквы слова за k_i (буквы в слове нумеруются с 0, т.е. если слово начинается с буквы a, то k_0=1).
4. Для каждой буквы определим число $h_i = k_i + N + i$.
5. В случае, если $h_i > 26$, $h_i = h_i \bmod 26$ (операция mod возвращает остаток от целочисленного деления).
6. Заменить каждую букву исходного слова на букву английского алфавита с номером h_i.

Применив свой алгоритм к некоторому осмысленному слову английского языка, Андрей получил **kwsbfxjye**. Определите минимальное N, при котором возможен такой результат, а также слово, к которому был применен алгоритм. В ответ запишите через пробел сначала число и затем слово.

Ответ: 10 algorithm

Решение:

Для начала заметим, что при всех N = 26x + y (x целое неотрицательное, $0 \le y \le 25$) при равных y, результат шифрования будет идентичен, т.к. величина h_i берется по модулю 26. Так как требуется определить минимальное N, будем рассматривать те, у которых $x = 0$. Значит, искомое N – число от 0 до 25 включительно.

Составим обратный алгоритм и проверим, что получится для всех возможных N:

```
    string s = "kwsbfxjye";
  for (int n = 0; n < 26; n++)
  {
    cout << n << " ";
    for (int i = 0; i < s.size(); i++)
    {
      int h_i = s[i] - 'a';
      h_i = h_i - i - n;
      if (h_i < 0) h_i += 26;
      cout << (char)(h_i + 'a');
    }
    cout << endl;
     }
```

Результатом работы алгоритма будет набор пар число-строка, где осмысленное слово будет только одно – algorithm, при n = 10, таким образом, ответ 10 algorithm.

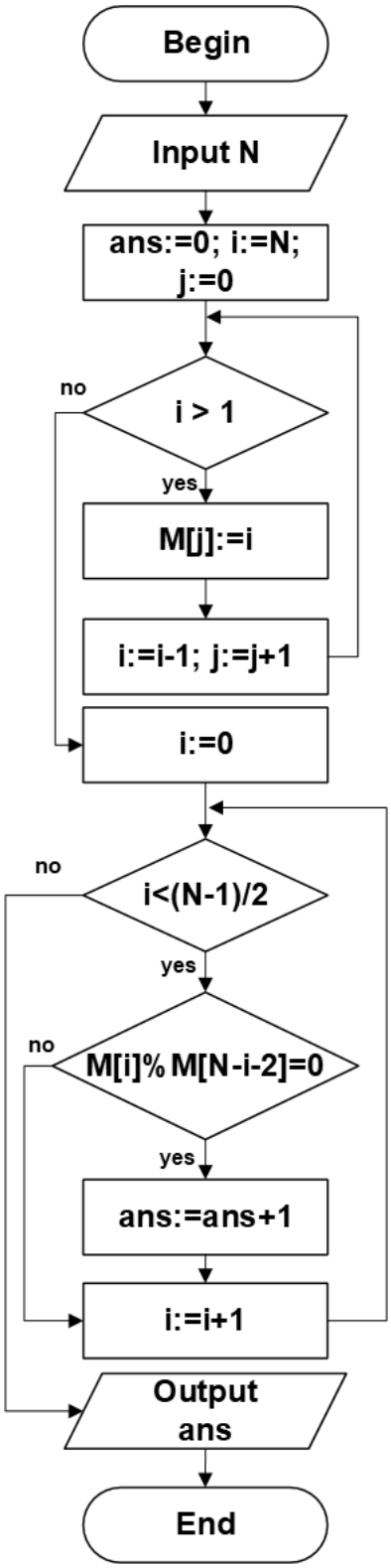
Пример реализации на Python:

```
from string import ascii_lowercase

word = 'kwsbfxjye'

for n in range(26):
    res = ''
    for i in range(len(word)):
        x = ascii_lowercase[(ascii_lowercase.index(word[i]) - i - n) % 26]
        res += x
    print(n, res)
```

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (2 балла)

[Считаем ноли]

Дана блок-схема алгоритма. M – массив размера N – 1. Операция A%B означает взятие остатка от деления A на B.



Алгоритм последовательно запустили со всеми натуральными значениями N от 100 до 199 включительно. Определите, при каком значении N был выведен наименьший результат (назовем его величиной A), а при каком – наибольший (назовем его величиной B). В случае, если величина A выводилась несколько раз – определите наименьшее из значений N, при которых это произошло. В случае, если величина B выводилась несколько раз – определите наибольшее из значений N, при которых это произошло. В ответе укажите через пробел 2 числа – сначала значение N для величины A и затем значение N для величины B.

Примечание: A/B означает операцию целочисленного деления A на B.

Ответ: 101 178

Решение:

Проэмулируем описанные в задаче действия, реализовав алгоритм на языке C++. Обратите внимания, для корректного определения N, при котором выводилась величина A, необходимо не обновлять значение в случае, если величина A встретилась повторно. В случае с величиной B – наоборот, каждый раз, когда выводимая величина равняется B, ответ необходимо обновить.

```
#include <iostream>
```

```
#include <vector>

using namespace std;
vector<int> numbers; //динамический массив чисел от N до 2

int main()
{
  int a = INT_MAX, b = 0, a_n = 0, b_n = 0;
  for (int n = 100; n < 200; n++)
  {
    int ans = 0; //счетчик количества 0
    numbers.clear();
    for (int i = n; i > 1; i--) //заполним динамический массив числами от текущего n до 2
    {
      numbers.push_back(i);
    }
    for (int i = 0; i < (n - 1) / 2; i++) //вычислим остатки
    {
      if (numbers[i] % numbers[n - i - 2]==0) ans++; //подчет числа 0
    }
    cout << ans << endl;
    if (ans < a) //определяем, нужно ли обновить величину A и ответ для нее
    {
      a = ans;
      a_n = n;
    }
    if (ans >= b) //определяем, нужно ли обновить величину B и ответ для нее
    {
      b = ans;
      b_n = n;
    }
  }
     cout << a_n << " " << b_n;
```

Пример реализации на Python:

```
max_ = -10 ** 9
min_ = 10 ** 9
max_n = min_n = 100
for n in range(100, 200):
    ans = 0
    m = [0] * n
    j = 0
    for i in range(n, 1, -1):
        m[j] = i
        j += 1
    for i in range((n - 1) // 2):
        if m[i] % m[n - i - 2] == 0:
            ans += 1
    if ans < min_:
        min_ = ans
        min_n = n
    if ans >= max_:
        max_ = ans
        max_n = n
print(min_n, max_n)
```

6. Телекоммуникационные технологии (1 балл) [Адреса и маски]

Маска сети для IPv4 адресации – это 4-х байтное число, которое делит IP адрес на адрес сети (первая часть) и адрес узла (вторая часть). У всех адресов одной IP-сети совпадают первые части и отличаются вторые. Для части IP адреса, соответствующей адресу сети, в маске сети содержатся двоичные единицы, а для части IP адреса, соответствующей адресу узла, в маске сети содержатся двоичные нули. Для записи масок сетей часто используется нотация, когда после IP-адреса через «/» указывается число бит, отводимых в маске под адрес сети. Например, для адреса 11.12.0.8 и маски 255.0.0.0 запись будет иметь следующий вид 11.12.0.8/8. Служебным адресом сети называют адрес, у которого все биты адреса узла (второй части) равны 0. Широковещательным адресом сети называют адрес, у которого все биты адреса узла (второй части) равны 1.

Известно, что широковещательный адрес некоторой сети – 192.121.87.255, а адрес 192.121.83.178 не принадлежит этой сети. Определите служебный адрес этой сети и ее маску. Если существует несколько возможных масок, укажите ту, у которой максимально количество узлов в сети.

Введите ответ на поставленную задачу в виде: A.B.C.D/M, где A, B, C, D – байты служебного адреса сети в десятичной системе счисления, а M – число бит, отводимых под адрес сети.

Ответ: 192.121.84.0/22

Решение:

Представим широковещательный адрес сети в двоичном формате: 192.121.87.255 = 11000000.01111001.01010111.11111111. По определению широковещательного адреса, первые 21 бит маски точно будут относиться к адресу сети, а не адресу узла. Теперь представим в двоичном формате второй известный нам адрес: 192.121.83.178 = 11000000.01111001.01010011. 10110010. Заметим, что первые 21 бит совпадают с широковещательным адресом, а 22-ой различается. Если бы к адресу сети относились первые 21 бит, то второй адрес принадлежал бы сети, а он не принадлежит. Значит, к адресу узла относится больше 21 бита. При этом, количество узлов будет максимально у той подсети, у которой минимальное количество бит относится к адресу сети. Минимальное число бит в маске больше 21 – это 22. Значит, наша искомая сеть имеет в двоичном формате вид 11000000.01111001.01010100.00000000/22, что в десятичном представлении записывается как 192.121.84.0/22.

7. Технологии обработки информации в электронных таблицах, технологии сортировки и фильтрации данных (3 балла)

[И тут XOR]

Дана таблица в режиме отображения формул:

	A	B	C	D
1		[redacted]		
2	0			=IF(B2<>C2; 1; 0)
3	1			
4	2			
5	3			
6	4			
7	5			
8	6			
9	7			
10	8			
11	9	=IF(B$1-SUM(B12:B$12) >= POWER(2; $A11); POWER(2; $A11); 0)		
12	10	=IF(B$1 >= POWER(2; $A12); POWER(2; $A12); 0)		

Формулу из ячейки B12 скопировали в ячейку C12. Формулу из ячейки B11 скопировали во все ячейки диапазона B2:C11. Формулу из ячейки D2 скопировали во все ячейки диапазона D2:D12.

В ячейку B1 записали число 1524, а в ячейку C1 некоторое натуральное число X. После чего в столбце D отобразились следующие значения:

	D
1	
2	1
3	1
4	1
5	1
6	0
7	0
8	1
9	1
10	0
11	1
12	0

Определите значение X.

Примечание: Таблица соответствия названий функций:

Название функции на английском	Название функции на русском
IF	ЕСЛИ
SUM	СУММ
POWER	СТЕПЕНЬ

Ответ: 1851

Решение:

Проанализируем формулы данной таблицы. Формулы в диапазоне B2:B12 отобразят в соответствующих ячейках B# 0, если A#-ый бит в двоичном представлении числа из ячейки B1 равен 0 и $2^{A\#}$, если A#-ый бит равен 1. Аналогично, формулы

в диапазоне C2:C12 отобразят в соответствующих ячейках C# 0, если A#-ый бит в двоичном представлении числа из ячейки C1 равен 0 и $2^{A\#}$, если A#-ый бит равен 1. Нумерация разрядов в двоичном представлении идет с 0, от младших разрядов к старшим. В столбце D будет отображаться 0, если некоторый бит совпадает в двоичных представлениях чисел B1 и C1 (т.е. если в обоих столбцах 0 или соответствующая степень) и 1, если бит различен. Таким образом, в столбце D будет записан в двоичном представлении результат операции B1 XOR C1. В ячейке D2 будет самый младший разряд, в ячейке D12 – старший. Таким образом, зная число в ячейке B1, мы можем получить его двоичное представление. По свойству операции XOR C1 = B1 XOR D (D – число, сформированное из битов столбца D). Т.е. X (значение в ячейке C1) вычисляется как 10111110100_2 XOR $01011001111_2 = 11100111011_2$. Т.е. X = 1851.

8. Технологии программирования (2 балла) [Тестирование мессенджера]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Перед выпуском VK Messenger'a разработчики из IT-компании «VK», как и положено, убеждаются в корректности работы приложения. Проверкой корректности работы систем занимаются тестировщики и QA-инженеры.

Часть функциональных тестов для тестирования смены ников выглядит следующим образом:

- генерируется случайный сценарий взаимодействия пользователей с приложением;
- в приложении симулируется выполнение этого сценария;
- проверяется корректность итогового состояния приложения.

Вам предлагается почувствовать себя в роли тестировщика и реализовать решение, обрабатывающее сценарий без взаимодействия с приложением. Результат обработки сценария затем будет сравниваться с итоговым состоянием приложения.

Каждый сценарий состоит из трех наборов событий.

Первый набор состоит из событий вида «в момент времени t_i поступил запрос регистрации нового пользователя с ID id_i и ником $handle_i$». Гарантируется, что ID уникальны для всех пользователей приложения. Если выбранный ник никем не занят, регистрация проходит успешно, иначе запрос на регистрацию отклоняется. Во втором случае пользователь с тем же ID может попробовать зарегистрироваться еще раз.

Второй набор описывает события вида «в момент времени t_i пользователь с ником $handle_{i,1}$ отправляет запрос на смену ника на $handle_{i,2}$». Гарантируется, что для каждого такого запроса ник $handle_{i,1}$ кому-то принадлежит. Если $handle_{i,2}$ уже занят каким-либо пользователем, запрос отклоняется, иначе пользователь успешно меняет ник. При успешной смене ника старый ник перестает ассоциироваться с каким-либо пользователем, пока кто-то снова его не займет.

Третий набор описывает события вида «в момент времени t_i пользователь с ником $handle_{i,1}$ отправляет сообщение пользователю с ником $handle_{i,2}$». Гарантируется, что и $handle_{i,1}$, и $handle_{i,2}$ на момент времени t_i соответствуют каким-то зарегистрированным пользователям.

Также гарантируется, что никакие два события не происходят в одно и то же время, то есть все t_i уникальны.

Вам даны описания всех трех наборов событий. Определите для каждого пользователя, сколько он всего отправил сообщений, и какому пользователю он отправил больше всего сообщений за весь сценарий.

Формат входных данных

В первой строке ввода дано единственное целое число T — количество сценариев, которое вам необходимо обработать ($1 \le T \le 100$).

Далее следуют T описаний сценариев. Описание каждого сценария начинается с пустой строки, после чего следуют три набора событий. В первой строке описания q-го набора (q от 1 до 3) дано единственное целое число n_q — количество событий в наборе, после чего следуют n_q строк в указанном ниже формате ($1 \le n_1+n_2+n_3 \le 1000$; $0 \le n_q$).

- События первого набора задаются в формате «t_i: REG id_i $handle_i$».
- События второго набора задаются в формате «t_i: CHANGE $handle_{i,1}$ $handle_{i,2}$».
- События третьего набора задаются в формате «t_i: SEND $handle_{i,1}$ $handle_{i,2}$».

Моменты событий t_i — целые числа от 1 до 10_9. Также все id_i — целые числа от 1 до 10_9, а $handle_i$ — строки из маленьких латинских букв длины не более 10.

Гарантируется, что в каждом наборе входных данных все t_i различны. Гарантируется, что успешно зарегистрированный пользователь не предпринимает попытки зарегистрироваться еще раз.

Формат выходных данных

Для каждого сценария выведите требуемую статистику для каждого пользователя.

В первой строке статистики выведите целое число q — количество зарегистрированных пользователей. В следующих q строках выведите статистику для каждого пользователя в порядке возрастания их ID в формате «<id> SENT <total> TOP <topcount$_{top}$> TO <id$_{top}$>», где total — суммарное количество отправленных пользователем сообщений, id_{top} — ID пользователя, получившего от него больше всего сообщений, а $count_{top}$ — само количество сообщений, отправленных пользователю id_{top}.

Если у некоторого пользователя есть несколько собеседников, получивших от него максимальное число сообщений, выведите в качестве id_{top} минимальный из их ID. Если пользователь не отправлял сообщения, считайте $count_{top}$ равным 0.

Пример

Стандартный ввод	Стандартный вывод

Стандартный ввод	Стандартный вывод
2 4 10: REG 1 sh 11: REG 2 sh 12: REG 3 qwerty 15: REG 2 shvetz 2 13: CHANGE sh shvetz 18: CHANGE shvetz sh 4 14: SEND shvetz qwerty 16: SEND shvetz shvetz 17: SEND shvetz qwerty 19: SEND qwerty sh 2 10: REG 1 a 11: REG 2 b 4 14: CHANGE a c 17: CHANGE b a 20: CHANGE c a 23: CHANGE c b 10 12: SEND a b 13: SEND b a 15: SEND b c 16: SEND b c 18: SEND a c 19: SEND a c 21: SEND c a 22: SEND a c 24: SEND a b 25: SEND a b	2 1 SENT 3 TOP 2 TO 3 3 SENT 1 TOP 1 TO 1 2 1 SENT 2 TOP 2 TO 2 2 SENT 8 TOP 8 TO 1

Решение

Задача на реализацию, требовалось сделать ровно то, что требуется по условию. Чтобы обработать события в правильном порядке, надо было выделить момент времени для каждого события, после чего отсортировать события по времени.

Выделить время в первом варианте можно было, удалив из первого слова в строке последний символ, а во втором — прочитав слово после «AT», после чего преобразовав полученное слово в integer.

Затем, после сортировки событий, необходимо их обработать. Будем хранить необходимую статистику по пользователям, а также информацию о том, какие ники и кем сейчас заняты. Для пользователя надо знать общее число отправленных сообщений total[id] и число отправленных сообщений в диалогах с каждым другим пользователем count[id1][id2]. Также, для каждого занятого на данный момент ника handle надо знать ID пользователя, который его занимает, users[handle].

Такую информацию можно хранить в переменных типов dict[int, int], dict[int, dict[int, int]] и dict[str, int] (Python). В Java и C++ ту же роль выполняют HashMap и unoredered_map.

Тогда обработка операций выглядит следующим образом (ниже приведен похожий на Python псевдокод, в реальном решении могут быть отличия из-за специфики выбранного языка):

1. регистрация: проверить, что ник не занят, и выдать его новому пользователю
id, handle <- событие
if handle not in users:
    users[handle] = id
    total[id] = 0
2. смена ника: проверить, что новый ник не занят, и поменять его
handle1, handle2 <- событие
id = users[handle1]
if handle2 not in users:
    users[handle2] = id
    users.pop(handle1)
3. сообщение: увеличить соответствующие счетчики в статистике
handle1, handle2 <- событие
id1, id2 = users[handle1], users[handle2]

total[id1]++
count[id1][id2]++

В самом конце надо для каждого пользователя вывести total[id] и самого частого собеседника по отправленным/полученным сообщениям. Для этого выпишем все пары «ключ-значение» из count[id] и отсортируем по значению, а при равном значении — по ключу. Требовалось также не забыть аккуратно обработать случай, когда пользователь не отправлял/не получал сообщения, в таком случае для него idtop равен min(users.values()).

9. Технологии программирования (4 балла)

[Устранение массива]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Дан массив a длины n. Каждый элемент массива — 0, 1 или 2. Известно, что нули находятся только строго в левой половине массива, то есть на индексах от 1 до ⌊n/2⌋. Все оставшиеся элементы могут быть равны только 1 или 2.

С массивом разрешается выполнять следующие два вида действий:

Заплатить две монеты и удалить из массива любое число a_i. После такого действия массив $[a_1,a_2,\ldots,a_k]$ превращается в $[a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_k]$.

Заплатить одну монету и заменить два стоящих рядом числа a_i и a_{i+1} на a копий числа a_{i+1}. То есть, возможны следующие преобразования:

$$[a_1,\ldots,a_{i-1},0,a_{i+1},\ldots,a_k] \to [a_1,\ldots,a_{i-1},a_{i+2},\ldots,a_k]$$
$$[a_1,\ldots,a_{i-1},1,a_{i+1},\ldots,a_k] \to [a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_k]$$
$$[a_1,\ldots,a_{i-1},2,a_{i+1},\ldots,a_k] \to [a_1,\ldots,a_{i-1},a_{i+1},a_{i+1},\ldots,a_k]$$

Иными словами, 0 можно удалить вместе со следующим числом, 1 можно удалить саму по себе, если она стоит не в конце массива, а 2 можно заменить на следующее за ней число.

Определите, какое минимальное число монет требуется заплатить, чтобы превратить данный массив в пустой. Обратите внимание, что ответ всегда существует, потому что можно просто n раз применить первую операцию.

Формат входных данных

В первой строке дано единственное целое число T — количество наборов входных данных ($1\le T\le 100$). Далее следуют T наборов входных данных.

Каждый набор входных данных начинается со строки, содержащей единственное число n — изначальную длину массива. В следующей строке через пробел перечислены целые числа a_1, ……, a_n — элементы массива ($0\le a_i\le 2$).

Гарантируется, что сумма длин массивов по всем наборам входных данных не превосходит 10^6.

Формат выходных данных

Выведите T целых чисел, каждое в своей строке — ответы на задачу для каждого набора входных данных в том порядке, в котором они даны во вводе.

Пример

Стандартный ввод	Стандартный вывод
4 7 0 2 0 1 1 2 2 5 1 1 1 1 1 11 0 1 2 2 0 1 2 2 1 2 1 10 1 2 1 2 0 2 2 2 2 1	5 6 9 11

Решение

Задача решалась некоторым подобием жадного алгоритма. Ниже мы приведем полное решение с доказательством, однако, чтобы придумать решение, не обязательно было его доказывать. Некоторые вещи, например, что (0, 2) → ∅ — самая «выгодная» операция, кажутся интуитивно понятными и без формального обоснования.

Решение задачи выглядит следующим образом.

1. Найдем самый правый 0 в массиве, после чего для всех элементов слева от него по очереди выполним замены (1, 0) → 0 или (2, 0) → (0, 0). Все единицы слева исчезли, все двойки превратились в нули.

2. Пусть у нас получилось ровно p нулей, а справа от них стоят q единиц и r двоек.

$$a = [\overbrace{0,0,\ldots,0}^{p},\overbrace{1,2,2,1,\ldots}^{q+r}]$$

Заметим, что

• от 1 на конце массива и от 2 в любом месте не избавиться никак, кроме как удалив их за две монеты или удалив их операцией (0, x) → ∅;

• от остальных единиц можно избавиться с помощью (1, x) → x;

• операция (2, 1) → (1, 1) «не выгодна», после придется избавляться от получившейся лишней единицы хотя бы за одну монету, тогда как можно было бы просто удалить исходную двойку за две монеты;

3. Если нули располагались только в левой половине массива, p 6 q+r. Поскольку операций вида (0, x) → ∅ может быть не больше, чем количество нулей, то есть p, справа могуть остаться «лишние» элементы, на которых нулей не хватит. Будем удалять «лишние» единицы с помощью операции (1, x) → x, пока это возможно.

4. После этого либо останется q + r = p, и можно будет за p монет удалить все с помощью нулей, либо будет все еще выполняться q + r > p, но справа не останется удаляемых за одну монету единиц.

5. В таком случае либо справа остались только двойки, либо двойки и одна единица на конце. Удалим p двоек с помощью (0, 2) → ∅, а на каждое из оставшихся чисел потратим по две монеты. Если на хвосте оставалась единица, то замены (2, 1) → (1, 1) → 1 тратят столько же монет, сколько и просто удаление одного числа.

Чтобы получить полный балл за задачу, требовалось посчитать p, q и r за линейное время, после чего посчитать ответ формулами, а не выполнять операции в явном виде — удаление из середины массива занимает много времени.

Дальше будет приведено строгое доказательство корректности этого решения. Посмотрим на следующую функцию:

Φ(a) = a.size() + sum(a) + 2C,

где C — количество потраченных монет для получения массива a из исходного. В конце и длина, и сумма элементов массива будут равны нулю, то есть будет выполняться Φ = 2C. Если требуется минимизировать C, то это то же самое, что и минимизировать Φ.

Заметим, что многие из разрешенных операций не меняют значение этой функции. А именно, замены (0, 0) → ∅, (1, x) → x и (2, 0) → (0, 0): C увеличивается на 1, а a.size()+sum(a) уменьшается на 2. Помимо этого, только две операции уменьшают Φ: (0, 1) → ∅ на 1 и (0, 2) → ∅ на 2, а остальные — увеличивают.

Посмотрим на ситуацию (p, q, r) после первого шага решения. Все операции в первом шаге — «бесплатные», они не меняют Φ. Заметим, что уменьшить Φ можно не больше, чем p раз, с помощью каждого из нулей в левой половине. При этом уменьшить Φ на два можно не больше, чем r раз, с помощью (0, 2) → ∅ с каждой двойкой в правой половине.

Если p 6 r, то лучшего результата добиться не получится, потому что каждая двойка справа от последнего нуля удаляется только с помощью (0, 2), либо за две монеты, то есть для минимизации Φ требуется максимизировать количество операций (0, 2) → ∅, что в данном случае сводится к получению как можно большего p, что мы и сделали. Оставшиеся шаги в решении — «бесплатное» удаление лишних единиц и вынужденное удаление всего оставшегося за две монеты.

Если p > r, то может показаться, что могло бы быть выгодно вместо каких-то операций (0, 1) → ∅, уменьшающих Φ на один, сделать больше операций (0, 2) → ∅ в левой части массива, получив p поменьше. Однако если сделать такую операцию в левой части массива, Φ уменьшается на 2, но и p уменьшается на 2, то есть теряются две потенциальные операции с ΔΦ > 1, которые в совокупности не хуже.

Таким образом, показанный алгоритм действительно минимизирует количество потраченных монет

Отборочный этап. Первый тур (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (1 балл) [123, 123…]

Записан ряд чисел в различных системах счисления: $123_4, 123_5, 123_6, \ldots 123_N$. Найдите сумму чисел в этом ряду, если N = 22. В ответ запишите одно десятичное число – полученную сумму ряда.

Ответ: 4332

2. Кодирование информации. Системы счисления (2 балла) [Половина в 2022]

Дано равенство:

$$0{,}2022_5 + 0{,}0068_{20} * 4D_{22} = X_N$$

Известно, что X - дробное число. В ответ напишите дробную часть числа X с ведущими нолями, если таковые имеются, если N = 12.

Пример записи ответа: если вы получили $X = 0.047_N$, запишите в ответ 047. Допускается использование цифр и заглавных латинских букв.

Ответ: 6

3. Кодирование информации. Количество информации. Кодирование текста (2 балла) [Hello, user!]

Женя разрабатывает дизайн стартовой страницы сайта. Он составил приветственный текст, начинающийся со слов Hello, user!, а после регистрации пользователя заменяет слово “user” в первом предложении текста на имя пользователя, указанное при регистрации. Текст Жени состоит из строчных и заглавных латинских букв, 12 различных символов пунктуации из набора { !, ?, ., @, -, :, /, *, (,), ;, , } и пробелов. Он решил кодировать этот текст, используя для каждого символа в нём минимально возможное, одинаковое для всех символов количество бит. На сколько бит изменится информационный объём, занимаемый текстом, если пользователь зарегистрируется с именем **rabbit**? В имени пользователя допускаются те же символы, что используются в остальном тексте. В ответе укажите целое число.

Ответ: 14

4. Кодирование информации. Объем данных (2 балла) [Воспоминания о Дэй-Сити]

Некий Би, имеющий кибернетические оптические протезы Сироки, вскоре после переезда в Дэй-сити получил в подарок новинку технологий - Носитель Информации. Он позволяет сохранять воспоминания об особо приятных событиях жизни. Носитель сохраняет сцену в памяти следующим образом: сохраняется то, что видел владелец и то, что он слышал. Сохранять запахи и ощущения технологии гномов пока что не позволяют. Сколько секунд события можно сохранить на Носитель Информации, если:

1. Глаза Би различают 65536 цветов, острота его зрения позволяет глазам формировать картинку 2048 на 1080 пикселей, а за секунду он видит 24 кадра. Известно, что на Носитель Информации записываются только коды цветов каждого кадра, при этом для записи кода каждого возможного цвета используется минимально возможное, одинаковое для всех цветов количество бит.
2. Би слушает обоими ушами с частотой дискретизации 32768 Гц и 65536 уровнями квантования. Известно, что на Носитель Информации записываются только коды уровней квантования сигнала для каждого уха, при этом для записи каждого возможного кода используется минимально возможное, одинаковое для всех кодов количество бит.
3. Би ещё не освоил сжатие данных, а потому не использует его.
4. Объем Носителя Информации - 1622 Мбайт

Примечание: 1 КБайт = 1024 байта, 1 МБайт = 1024 КБайта.

Ответ: 16

5. Основы логики. Анализ логических функций (1 балл) [Всё перепуталось]

Даны три таблицы истинности:

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Таблица A

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Таблица B

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Таблица C

И пять логических функций:

1. $(a \to b) \wedge (b \to c) \wedge (c \to a)$
2. $(a\ XOR\ b) \vee (b\ XOR\ c) \wedge (c\ XOR\ a)$
3. $(a \to b) \to c$
4. $(a \equiv b) \wedge c \vee (a \equiv c) \wedge b \vee (b \equiv c) \wedge a$
5. $a \wedge b \vee b \wedge c \vee c \wedge a$

Определите, каким функциям принадлежат таблицы A, B и C. В ответ запишите подряд 3 цифры - номера функций для таблиц A, B и C, в указанном порядке.

Пример записи ответа: 123

Ответ: 341

6. Основы логики. Упрощение логического выражения (2 балла) [Трое из ларца]

Упростите логическое выражение или укажите его результат (при его однозначности). Результат упрощения может содержать только операции инверсии, конъюнкции и дизъюнкции и не должен содержать скобок.

$$\overline{(A \to B)} \vee (B \equiv C) \vee (C\ XOR\ A)$$

*Комментарий по вводу ответа: операнды вводятся большими **латинскими** буквами; логические операции обозначаются, соответственно, как **not**, **and** и **or**.*

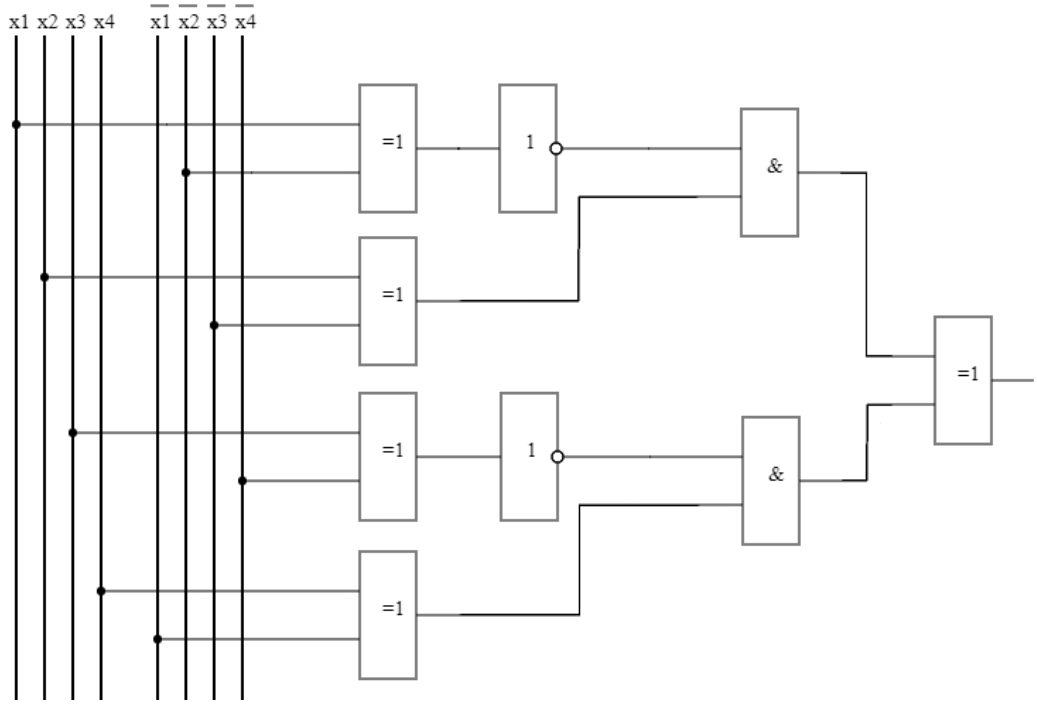
При однозначном ответе – истинный ответ обозначается как 1, а ложный как 0.

Пример записи ответа: A or B and not C

Ответ: A or not B or C || A or C or not B || not B or A or C || not B or C or A || C or A or not B || C or not B or A

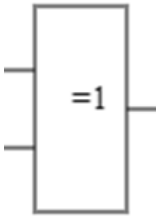
7. Основы логики. Синтез выражения по логической схеме (3 балла) [Опять поксорились!]

Дана схема логической функции F от четырёх переменных:



Сколько существует различных наборов значений переменных x1, x2, x3, x4 таких, что в результате будет получено значение 1 (истина)? В ответе укажите число.

Примечание. На схеме использованы следующие обозначения логических операторов:

Конъюнкция	Инверсия	Исключающее ИЛИ
		

Ответ: 4

8. Алгоритмизация и программирование. Формальный исполнитель (3 балла) [Кузнечик и тетраэдр]

Кузнечик прыгает по неотрицательной части числовой оси. Он начинает движение в точке (0) и прыгает согласно броскам четырехгранного кубика (четырехгранный кубик имеет форму тетраэдра, в результате броска может выпасть одно из чисел { 1, 2, 3, 4 }). Он использует следующие правила для определения точки, в которую он прыгнет:

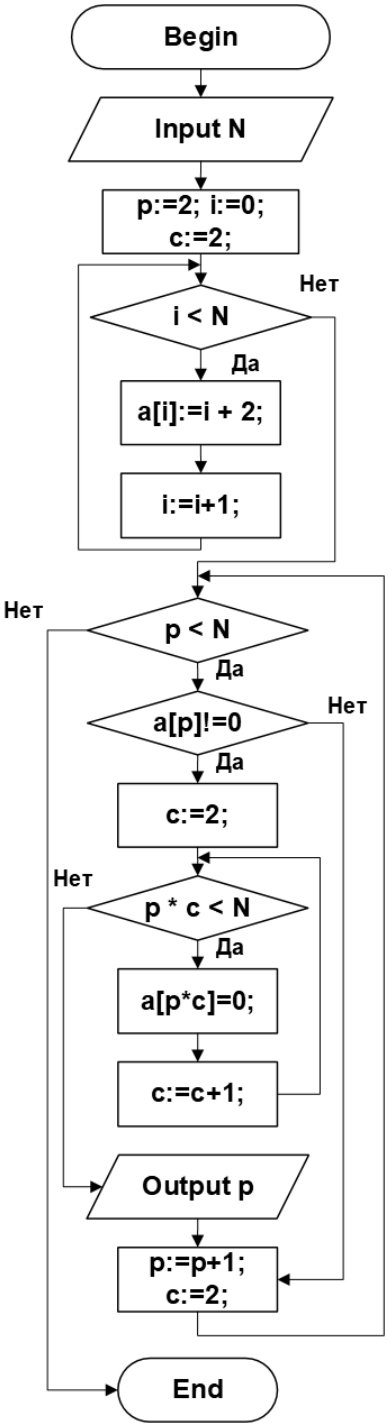
1. Если это нечетный прыжок (кузнечик нумерует прыжки с единицы), то он бросает кубик и прыгает вперёд на выпавшее число. Например, находясь в точке (2) и выбросив на кубике 3, он прыгнет вперёд на 3 и окажется в точке (5).
2. Если это четный прыжок, то он бросает кубик и прыгает назад на выпавшее число, если может совершить такой прыжок не выходя за пределы неотрицательной части числовой оси. Если прыжок он совершить не может, он бросает кубик заново и повторяет это до тех пор, пока не совершит удачный прыжок. Например, находясь в точке (2) и выбросив на кубике значение (3) совершить прыжок он не сможет. А выбросив значение 1, он прыгнет в точку (1).

Сколько существует точек, таких, что кузнечик может оказаться в них после 15 прыжков?

Ответ: 25

9. Алгоритмизация и программирование. Блок-схема, обратная задача (2 балла) [Сквозь решето]

Дана блок-схема алгоритма:



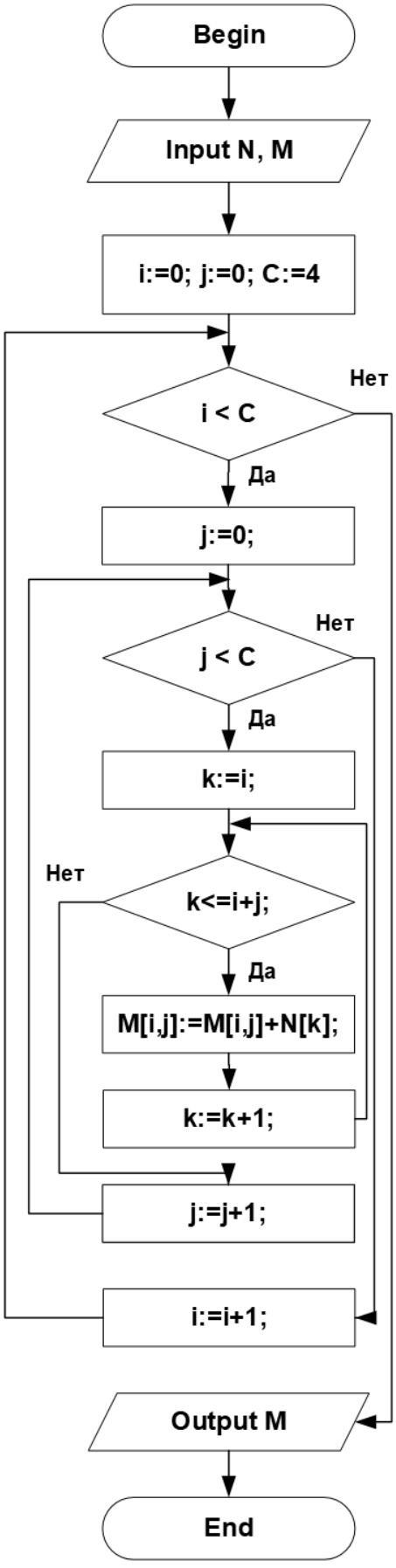
На вход подали натуральное число N. Известно, что в результате выполнения алгоритма было выведено 13 чисел. Найдите минимальное N, при котором это возможно.

Ответ: 42

10. Алгоритмизация и программирование. Блок-схема, массивы (3 балла)

[Суммы цифр]

Дана блок-схема алгоритма, обрабатывающего целочисленную матрицу, размером 4 на 4 элементов:



На вход подали целое положительное число N и матрицу M, заполненную только нулевыми значениями:

На выходе получили следующую матрицу M:

2	5	10	18
3	8	16	23
5	13	20	24
8	15	19	25

При каком минимальном значении N это возможно? В ответе укажите целое положительное число. Если такого числа не существует, в ответе напишите NULL.

Примечания:
1. При обращении к элементам матрицы первый индекс означает номер строки, а второй – номер столбца. Нумерация производится от 0.
2. Операция N[k] означает операцию взятия k-ой цифры числа. Цифры числа нумеруются слева направо с 0.

Ответ: 2358746

Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Электронные таблицы. Адресация ячеек и вычисления (3 балла) [Паскаль на базе Фибоначчи]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D
1	1	1	=A$1+B$1	
2	1	=A2+B1		
3	=$A1+$A2			
4				
5				
6				
7				
8				

Ячейку C1 скопировали во все ячейки диапазона D1:*1, ячейку A3 во все ячейки диапазона A4:A#, ячейку B2 во все ячейки диапазона B2:*#, где * - адрес некоторого столбца, # адрес некоторой строки, а *# адрес некоторой ячейки на пересечении строки # и столбца *. После чего посчитали в диапазоне B2:*# количество ячеек с числами, удовлетворяющими неравенству $x \le @@ \le y$, где @@ - адрес ячейки, а x и y – различные натуральные числа. Результаты этих вычислений разместили в таблицу:

x	y	Число ячеек
5000	7000	5
6000	9000	7
7000	11000	2
8000	13000	6
9000	15000	6
10000	17000	6
11000	19000	6
12000	21000	6
13000	23000	4
14000	25000	6
15000	27000	5
16000	29000	5
17000	31000	5
18000	33000	5
19000	35000	7
20000	37000	5
21000	39000	5
22000	41000	5
23000	43000	5
24000	45000	5

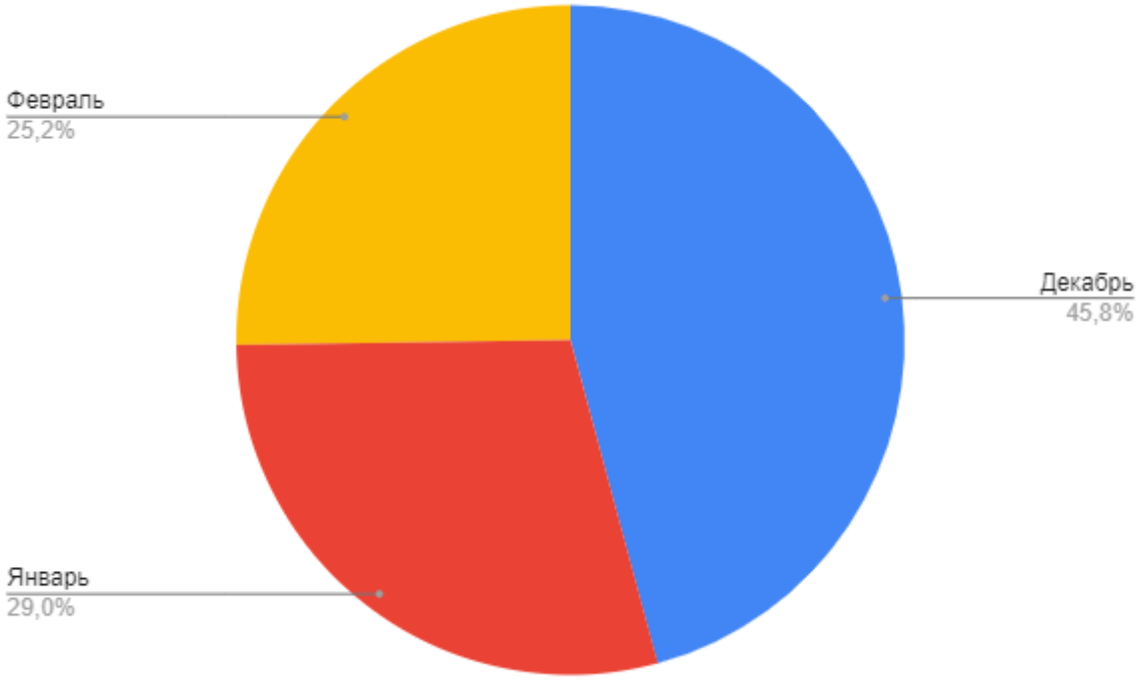
Определите, какой адрес ячейки был зашифрован как *# (т.е. адрес второй ячейки, определяющей диапазон B2:*#). В ответ напишите без пробела сначала заглавную латинскую букву, обозначающую номер столбца, затем число, обозначающее номер строки.

Ответ: L12

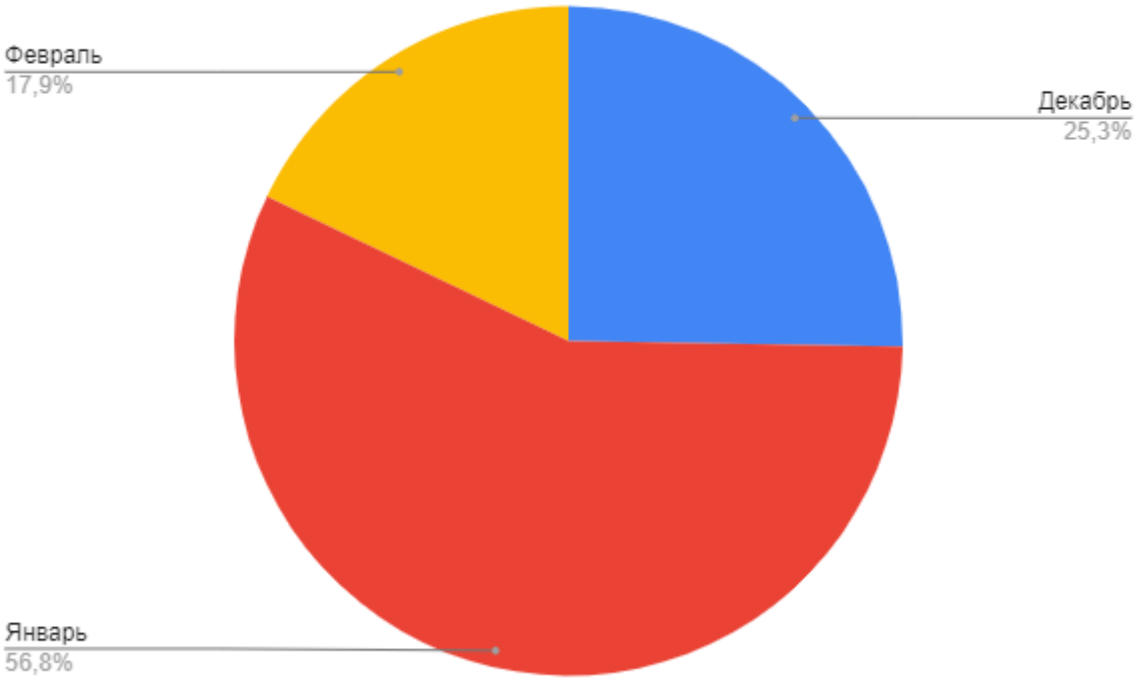
2. Электронные таблицы. Графики и диаграммы (2 балла)

[Зимний отпуск]

Сотрудники некоторой компании брали отпуска зимой. Распределение по месяцам выглядит следующим образом:



В прошлом году аналогичное распределение выглядело так:



Известно, что в феврале прошлого года отпуск брали 17 сотрудников, а общее число сотрудников, взявших отпуск зимой, с прошлого года увеличилось на 12. Определите, сколько сотрудников взяли отпуск в декабре этого года.

Примечание: в случае получения нецелого ответа, округлите число до ближайшего целого.

Ответ: 49

3. Сортировка и фильтрация данных (2 балла)

[Сравни со стабильной]

Стабильная сортировка — сортировка, которая не меняет относительный порядок сортируемых элементов, имеющих одинаковые ключи, по которым происходит сортировка. Нестабильная сортировка - сортировка, которая не гарантирует относительный порядок сортируемых элементов, имеющих одинаковые ключи, по которым происходит сортировка.

Приведем пример:

Key	Value
3	Blue

Key	Value
1	Red

Key	Value
1	Red

2	Green
1	Red
2	Yellow

Исходный порядок объектов

2	Yellow
2	Green
3	Blue

Объекты отсортированы по ключу Key по возрастанию нестабильной сортировкой. Относительный порядок элементов с ключом 2 изменился.

2	Green
2	Yellow
3	Blue

Объекты отсортированы по ключу Key по возрастанию стабильной сортировкой. Относительный порядок элементов с ключом 2 остался прежним.

В магазине рюкзаков составили сравнительную таблицу по четырем параметрам: цвет, материал, число карманов и размер. Получили следующий набор данных:

gray	waterproof canvas	1	M
white	smooth leather	3	L
red	textured leather	2	M
white	waterproof canvas	3	M
green	textured leather	1	M
black	textured leather	2	L
brown	canvas	3	S
black	textured leather	1	S
green	canvas	1	M
blue	smooth leather	3	L
gray	smooth leather	3	L
gray	waterproof canvas	1	M
gray	smooth leather	1	S
brown	textured leather	3	L
yellow	textured leather	2	M

После чего строки данной таблицы отсортировали нестабильной сортировкой сначала по цвету (в алфавитном порядке, по возрастанию), а затем по числу карманов (по убыванию). В результате этого таблица приобрела следующий вид:

black	textured leather	2	L
black	textured leather	1	S
blue	smooth leather	3	L
brown	canvas	3	S
brown	textured leather	3	L
gray	smooth leather	3	L
gray	smooth leather	1	S
gray	waterproof canvas	1	M
gray	waterproof canvas	1	M
green	canvas	1	M
green	textured leather	1	M
red	textured leather	2	M
white	smooth leather	3	L
white	waterproof canvas	3	M
yellow	textured leather	2	M

Сколько строк не совпало бы, если бы таблицу отсортировали по тем же критериям (сначала по цвету (в алфавитном порядке, по возрастанию), а затем по числу карманов (по убыванию)), но стабильной сортировкой?

В ответ напишите одно число – количество строк таких, что их содержимое различно в приведенном результате нестабильной сортировки и результате стабильной сортировки этой же таблицы.

Ответ: 4

4. Поиск и фильтрация данных (1 балл)

[Считаем призёров]

Результаты участников некоторой олимпиады оценивались из 100 баллов. По итогам, школьники, набравшие 90 и более баллов, награждались дипломами победителя олимпиады, а школьники, набравшие более 74, но менее 90 баллов, награждались дипломами призеров.

В олимпиаде принимали участие школьники 9 и 10 классов, каждый из них набрал по итогам целое неотрицательное число баллов, не превышающее 100.

Организаторы сделали несколько запросов к базе данных с результатами и выяснили следующее:

1. В олимпиаде 79 победителей.
2. 75 десятиклассников получили дипломы.
3. 68 девятиклассника не стали победителями.
4. Среди участников 71 не получили никакого диплома.
5. Десятиклассников, не являющихся призерами – 89.

Определите число призеров олимпиады, если известно, что среди девятиклассников 31 победитель. В ответ запишите одно число – количество призеров.

Ответ: 65

5. Телекоммуникационные технологии (2 балла)
[Кто скрывается под маской?]

Маска сети для IPv4 адресации – это 4-х байтное число, которое делит IP-адрес на адрес сети (первая часть) и адрес узла (вторая часть). У всех адресов одной IP-сети совпадают первые части и отличаются вторые. Для части IP-адреса, соответствующей адресу сети, в маске сети содержатся двоичные единицы, а для части IP-адреса, соответствующей адресу узла, в маске сети содержатся двоичные нули. Для записи масок сетей часто используется нотация, когда после IP-адреса через «/» указывается число бит, отводимых в маске под адрес сети. Например, для адреса 11.12.0.8 и маски 255.0.0.0 запись будет иметь следующий вид 11.12.0.8/8. Служебным адресом сети называют адрес, у которого все биты адреса узла (второй части) равны 0.

Некоторая сеть задана в виде служебного адреса этой сети и маски: 192.168.160.0/19. Выберите все адреса узлов, принадлежащих этой сети.

1. 192.168.52.17
2. 192.168.183.92
3. 192.168.89.83
4. 192.168.107.170
5. 192.168.233.249
6. 192.168.191.42
7. 192.168.212.83
8. 192.168.2.179
9. 192.168.171.157
10. 192.168.217.227
11. 192.168.219.81
12. 192.168.146.128

Ответ: 2 6 9

6. Операционные системы (3 балла)
[Round Robin на задумчивом процессоре]

На некотором вычислительном узле с одним процессором параллельная обработка нескольких процессов происходит согласно алгоритму Round Robin – процессам поочередно предоставляется возможность монопольно выполняться, однако каждый процесс может занимать процессор не более чем заданное число квантов времени. Например, если процессор предоставляется на 3 кванта, а процессу A необходимо 6 квантов чтобы завершиться, то процесс A будет выполняться в течении 3 квантов, затем возможность будет передана другому процессу. Оставшиеся необходимые 3 кванта процессорного времени процесс A получит тогда, когда очередь дойдет до него повторно. Процесс B, для выполнения которого нужно всего 2 кванта времени, получив возможность выполняться, выполнится целиком и очередь сразу же (не дожидаясь истечения всех 3 квантов) перейдет следующему процессу. Временем на переключение процессов на процессоре в рамках этой задачи пренебрегаем. Рассматриваемый вычислительный узел может параллельно обрабатывать не более 3 различных процессов. Таким образом, сначала возможность выполняться предоставляется процессу 1, затем процессу 2, затем процессу 3, затем снова процессу 1 и так далее.

Каждый процесс может находиться в двух состояниях – ожидает выполнения (т.е. выполняется условие «процесс создан, не завершен, но в данный квант он не выполняется») или процесс выполняется в данный момент на процессоре.

На вычислительном узле работают три приложения, каждое из которых регулярно создает новый процесс по истечении одного кванта после того, как был завершен предыдущий процесс этого приложения. На выполнение каждому из процессов требуется 6 квантов вне зависимости от того, каким приложением был создан этот процесс.

На момент начала первого кванта (кванты нумеруются с 1), созданы процессы всех трех приложений (приложение 1, приложение 2 и приложение 3), обработка начинается с процесса приложения 1, затем будет выполнятся процесс приложения 2, затем процесс приложения 3, затем опять процесс приложения 1 и т.д. Процессор предоставляется каждому процессу на 4 кванта. Определите, процессы каких приложений будут выполняться на 101-м, 201-м и 301-м квантах. В ответ запишите через пробел 3 числа – номера соответствующих приложений.

Ответ: 3 1 1

7. Технологии программирования (3 балла)
[Плейлисты]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	3 секунды
Ограничение по памяти	256 мегабайт

Аркадий гуляет по городу и слушает музыку. Всего на телефоне у Аркадия k плейлистов, каждый из которых состоит из одной или более песен. Каждая песня может лежать в нескольких плейлистах, то есть наборы песен в плейлистах могут пересекаться.

Когда текущий плейлист заканчивается или Аркадию надоедает его слушать, он переключается на другой. При переключении на другой плейлист у Аркадия есть две опции:

- Continue: продолжить слушать плейлист с того места, на котором он остановился при прошлом прослушивании (либо с начала, если этот плейлист еще не был включен ранее);

- Start: начать прослушивание плейлиста с первой песни.

Гарантируется, что Аркадий не переключается между плейлистами, не дослушав до конца текущую песню. Иными словами, переключение между плейлистами происходит ровно в момент между окончанием одной песни и началом другой.

Вам даны списки песен в каждом плейлисте с указанием их длительности. Помимо этого известно, в каком порядке и сколько времени Аркадий слушал конкретные плейлисты. Ваша задача — для каждой песни определить сколько раз она прозвучала за указанный период времени. Песня однозначно задается названием и исполнителем.

Формат входных данных

В первой строке ввода дано одно целое число t — количество тестовых наборов ($1 \leqslant t \leqslant 50$). Далее следуют t тестовых наборов. Перед каждым тестовым набором находится пустая строка. Описание одного тестового набора состоит из описания плейлистов и описания действий Аркадия.

Первая строка описания плейлистов содержит одно целое число k — количество плейлистов на телефоне Аркадия ($1 \leqslant k \leqslant 50$). Затем следуют описания плейлистов. Описание i-го плейлиста начинается со строки в формате ("<Название плейлиста>" n_i), содержащей через пробел название плейлиста в кавычках и количество песен в плейлисте n_i ($1 \leqslant n_i \leqslant 100$). Далее расположены n_i строк в формате ("<Название песни>" by "<Исполнитель>" for <MM:SS>), где <MM:SS> — длительность песни (минуты и секунды, по два символа каждая). Обратите внимание, что каждая строка с песней начинается с пробела.

Названия плейлистов, песен и исполнителей — строки из букв латинского алфавита и пробелов длины не более 2020, и не могут начинаться с пробела. Названия всех плейлистов различны. Гарантируется, что если песня входит в несколько плейлистов, то у нее указана одинаковая длительность в каждом из них.

Первая строка описания действий Аркадия содержит целое число q — количество самих действий ($1 \leqslant q \leqslant 100$). В следующих q строках заданы сами действия в формате (<Действие> "<Название плейлиста>" for <MM:SS>), где <Действие> может быть либо «Continue», либо «Start», а MM:SS — время прослушивания плейлиста после запуска в минутах и секундах.

Гарантируется, что если текущий плейлист заканчивается, то это либо последний плейлист на сегодня, либо Аркадий сразу же переключается на какой-то другой.

Формат выходных данных

Для каждого из тестовых наборов выведите список песен, прозвучавших хотя бы один раз.

Список песен должен начинаться со строки, содержащей одно целое число m — количество песен в нем, после чего должны следовать m строк в формате ("<Название песни>" by "<Исполнитель>" times <X>), где <X> — количество раз, которое эта песня прозвучала.

Песни в списке должны следовать в алфавитном порядке по исполнителю, а для одного исполнителя — в алфавитном порядке по названию.

Пример

Стандартный ввод	Стандартный вывод
3 1 "One song I listen to" 1 "The Handler" by "Muse" for 04:33 2 Start "One song I listen to" 04:33 Start "One song I listen to" 04:33 3 "Metal" 1 "Unforgiven" by "Metallica" for 06:20 "Pop" 1 "Firework" by "Katy Perry" for 03:49 "All" 2 "Unforgiven" by "Metallica" for 06:20 "Firework" by "Katy Perry" for 03:49 3 Continue "Metal" 06:20 Continue "Pop" 03:49 Continue "All" 10:09 2 "My Favourites" 3 "Parable" by "Mako" for 03:29 "Post Blue" by "Placebo" for 03:11 "Overglow" by "Adam Lambert" for 03:32 "Chill" 3 "Parable" by "Mako" for 03:29	1 "The Handler" by "Muse" times 2 2 "Firework" by "Katy Perry" times 2 "Unforgiven" by "Metallica" times 2 4 "Overglow" by "Adam Lambert" times 1 "Coyote" by "Mako" times 1 "Parable" by "Mako" times 3 "Post Blue" by "Placebo" times 1

Стандартный ввод	Стандартный вывод
"Coyote" by "Mako" for 03:56 "Interlude" by "Enter Shikari" for 00:56 4 Start "My Favourites" 06:40 Start "Chill" 03:29 Continue "My Favourites" 03:32 Start "Chill" 07:25	

8. Технологии программирования (5 баллов) [Минимальное покрытие]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Даны n точек на плоскости, i-я точка имеет координаты (x_i,y_i). Никакие две точки не находятся на одной вертикальной или горизонтальной прямой, то есть все x_i различны и все y_i различны.

Требуется покрыть эти точки двумя непересекающимися листами бумаги наименьшей суммарной площади. Стороны листов бумаги должны быть параллельны осям координат. Точка, лежащая на границе листа бумаги, считается покрытой им.

Определите, прямоугольниками какой минимальной суммарной площади можно покрыть все точки заданного множества.

Формат входных данных

В первой строке ввода дано одно целое число t — количество тестовых наборов ($1 \leqslant t \leqslant 30$). Далее следуют t тестовых наборов.

Первая строка входных данных тестового набора содержит одно целое число n — количество точек на плоскости ($1 \leqslant n \leqslant 10^4$).

В i-й из следующих n строк через пробел даны два целых числа x_i и y_i — координаты i-й точки ($|x_i|,|y_i| \leqslant 10^6$). Гарантируется, что все x_i различны и все y_i различны.

Формат выходных данных

Выведите t целых чисел, каждое в своей строке — i-е число должно быть равно минимальной площади прямоугольников, которыми можно покрыть точки из i-го тестового набора.

Если точки могут быть покрыты двумя вырожденными прямоугольниками (высоты или ширины 0), выведите 0.

Пример

Стандартный ввод	Стандартный вывод
4 2 0 0 1 1 3 0 0 1 1 2 2 4 0 4 2 1 5 -1 3 5 7 -5 9 0 3 1 -6 2 2 -7 -3 -1 6 8 8	0 1 9 115

Задания для 5–8 класса

Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла) [Неизвестные цифры]

Дана последовательность из 4 чисел:

36_{10} $N3_8$ $2M2_4$ $10L111_2$

Известно, что каждое следующее число больше, чем предыдущее, и определяется единственным способом. Найдите недостающие цифры.

В ответ укажите значения цифр N, M и L через пробел в указанном порядке.

Ответ: 5 3 1

Решение:

Составим систему уравнений, используя формулу развернутой записи числа:

$$\begin{cases} N3_8 = N * 8 + 3; N \in [1;7] \\ 2M2_4 = 2 * 16 + M * 4 + 2 = 34 + M * 4;\ M \in [0;3] \\ 10L111_2 = 1 * 32 + L * 8 + 1 * 4 + 1 * 2 + * 1 * = 39 + L * 8;\ L \in [0;1] \end{cases}$$

Переменная L может принимать только два значения, возьмём меньшее из них.

Пусть L = 0. Получаем следующее неравенство: 34 + 4M <39. Значит число M может принимать только два значения: 0 и 1. Также у нас есть неравенство 36 < 34 + 4M. Следовательно, M = 1. Остается два неравенства:

$$\begin{cases} 36 < 8N + 3 \\ 8N + 3 < 38 \end{cases}$$

Не существует такого натурального N, которое бы являлось решением данной системы.

Значит L не может быть равно 0 и равно 1. Рассмотрим следующее неравенство:

36 < 8N + 3. Получаем, что N > 4. Возьмем наименьшее из возможных значение N = 5. Получаем неравенство 34 + 4M > 43. Отсюда следует, что M > 2. Следовательно, M = 3.

Теперь найдем правильное значение N:

$$\begin{cases} 36 < 8N + 3 \\ 8N + 3 < 46 \end{cases}$$

Получаем что N = 5. Проведём проверку полученных значений:

$53_8 = 43_{10}; 232_4 = 46_{10}; 101111_2 = 47_{10}$.

36 < 43 < 46 < 47

2. Кодирование информации. Шифрование (2 балла) [Фруктовый шифр]

Настя очень любит шифрование, и, зная это, мама предложила ей обмениваться сообщениями с помощью последовательностей яблок (Я), мандаринов(М) и груш(Г), выложенных на столе. Комбинации фруктов, соответствующие буквам, представлены в таблице.

A	B	C	D	E	F	G	H	I
ГЯГ	МЯ	ГГ	ЯМГ	ЯЯГ	ГМ	ГЯ	ГМЯ	МММ

Буквы послания, зашифрованные с помощью фруктов, выкладываются на стол подряд, в линию, без пропусков и каких-либо разделителей.

В один прекрасный день Настя, заходя на кухню, увидела, что на столе ее ждет зашифрованное фруктами послание. Однако вместе с Настей на кухню забежала ее младшая сестра, не посвященная в тонкости договоренностей между Настей и мамой. Она схватила несколько фруктов из начала и конца последовательности и убежала с ними к себе в комнату. Настя, к своему величайшему сожалению, не успела запомнить, что это были за фрукты и сколько их было.

Когда Настя подошла после этого к кухонному столу, на столе фрукты лежали в следующем порядке:

ММГГМЯГЯГММЯГМЯГГЯМ

Расшифруйте послание, которое мама оставила для Насти, если известно, что оно состоит из 9 букв, а также гарантируется, что младшая сестра забрала один или несколько подряд идущих фруктов из начала последовательности, и один или несколько подряд идущих фруктов из конца последовательности.

Ответ запишите заглавными латинскими буквами. Если возможно несколько расшифровок сообщения, в качестве ответа можно взять любую.

Примечание: зашифрованное послание может быть бессмысленным набором букв.

Ответ: ICBGFBHCD

Решение:

При расшифровке сообщения нельзя ориентироваться на начало или конец последовательности, так как они неизвестны. Значит, необходимо найти какую-то комбинацию в середине послания, которая расшифровывается совершенно однозначно, чтобы в дальнейшем отталкиваться от нее.

Например, можно заметить, что два мандарина (такая комбинация есть в середине) подряд не идут в кодировке ни в одной из букв, кроме «I», но это точно не она, так как там 3 «М» подряд. Единственный вариант, вследствие которого эти два мандарина могли оказаться рядом – это конец буквы «F» и начало буквы «B». Выделив комбинации для этих букв, можно начать расшифровывать «в стороны», и крайние буквы (из-за ограничения на буквы), будут найдены однозначно.

Примечание: расшифровку можно начать и с другой однозначно расшифровываемой комбинации из середины послания, если участник заметит другую.

Рассмотрим расшифровку по шагам.

1. Комбинация фруктов «ГММЯ» в середине расшифровывается однозначно: «ГМ» – «F», «МЯ» – «B».

ММГГМЯГЯ**ГММЯ**ГМЯГГЯМ – …**FB**…

Здесь и далее жирным будут выделяться буквы, которые рассматриваются на этом шаге расшифровки, подчеркнутым сплошной линией – буквы, определенные ранее однозначно, подчеркнутые «зигзагом» - буквы, которые мы определили предположительно.

2. Начнем расшифровку вправо. Какая буква может начинаться с «ГМ»? Это либо «F», либо «H». Предположим, что это «F».

ММГГМЯГЯГММЯ**ГМ**ЯГГЯМ – …FB**F**…

Но тогда, следующая за новой «F» буква должна начинаться с «ЯГ»:

ММГГМЯГЯГММЯГМ**ЯГ**ГЯМ – …FB**F?**…

С сочетания фруктов «ЯГ» не начинается ни одна буква, значит, наше предположение насчет того, что «ГМ» – это «F» – неверно, и на самом деле это все-таки буква «H». Теперь однозначно определено три буквы.

ММГГМЯГЯГММЯГМЯГГЯМ – …FBH…

3. Продолжим расшифровку вправо. Сочетание фруктов «ГГ» можно расшифровать только как «C».

ММГГМЯГЯГММЯГМЯ**ГГ**ЯМ – …FBH**C**…

4. Незадействованными в расшифровке осталось 2 фрукта – «ЯМ». Учитывая, что это начало буквы, можем однозначно определить, что это должна быть буква «D» (никакая другая буква на «ЯМ» не начинается) – таким образом получили, что справа не хватает одной груши.

ММГГМЯГЯГММЯГМЯГГ**ЯМ**Г – …FBHC**D**…

Можно предположить, что справа также лежали и другие фрукты, обозначавшие еще одну букву, но пока не будем больше ничего добавлять – возможно букв, полученных после расшифровки левой части, будет достаточно (известно, что букв должно быть ровно 9), а один фрукт справа уже добавлен.

ММГГМЯГЯГММЯГМЯГГЯМГ – …FBHCD…

5. Переходим к расшифровке левой части послания, также будем отталкиваться от центра. На «ГЯ» заканчивается только одна буква – «G».

ММГГМЯ**ГЯ**ГММЯГМЯГГЯМГ – …**G**FBHCD…

6. На «МЯ» заканчиваются две буквы – «H» и «B». Предположим, что это буква H. Но тогда буква, стоящая слева от «H», должна заканчиваться на «МГ», - а таких букв нет.

М**МГ**ГМЯГЯГММЯГМЯГГЯМГ – …**?**HGFBHCD…

Значит, наше предположение неверно и слева от буквы «G» стоит буква «B».

ММГГМЯГЯГММЯГМЯГГЯМГ – …BGFBHCD…

7. «ГГ» расшифровывается однозначно – это буква «C».

ММ**ГГ**МЯГЯГММЯГМЯГГЯМГ – …**C**BGFBHCD…

8. Осталось два мандарина – на последовательность «ММ» может заканчиваться только одна буква – «I» – значит, слева должен лежать еще один мандарин.

М**ММ**ГГМЯГЯГММЯГМЯГГЯМГ – …**I**CBGFBHCD…

Итак, мы получили:

ММММГГМЯГЯГМЯГМЯГГЯМГ – ICBGFBHCD

Так как расшифрованных букв уже 9 (ровно столько, сколько и должно было быть по условию), то ни слева, ни справа, фруктов больше быть не должно, последовательность расшифровывалась однозначно.

3. Комбинаторика (1 балл)
[Двоичные последовательности]

Даша получила письмо с зашифрованным сообщением, состоящим из нулей и единиц, от подруги Маши: «? 0 1 0 1 1 1 0» (символ на месте «?» Даша, к сожалению, не смогла разобрать). Даша знает, что придуманный Машей алгоритм шифрования состоит из одного простого действия: одновременной замене двух соседних символов на противоположные (символы в последовательности нумеруются с 1). Например, если дана последовательность 01 00 00 00 00, то совершение действия для пары (1,2) превратит последовательность в 10 00 00 00 00. Для того, чтобы зашифровать сообщение, это действие производится несколько раз в определенном порядке над разными парами цифр последовательности.

Маша записала действия, которые она совершила во время шифрования, на карточках и приложила все карточки к письму. Но, к сожалению, они перемешались, и Даша их выложила перед собой в случайном порядке: (1,2), (3,4), (4,5), (1,2), (7,8), (6,7)

Так как Даша не знала правильный порядок операций и нечитаемый символ, она не могла проделать все операции в точности в обратном порядке и таким образом восстановить начальную последовательность (по Машиной задумке расшифровка должна была происходить именно так).

Но Даша не отчаялась: она решила найти все возможные последовательности, из которых, с помощью операций, записанных на карточках, могла была быть получена та, что находилась перед ней.

Помогите Даше и ответьте на вопрос, сколько различных последовательностей можно получить, расшифровывая данную? В ответе укажите целое число.

(Под данной последовательностью понимается «? 0 1 0 1 1 1 0», где на месте «?» может стоять как «1», так и «0»)

Ответ: 2

Решение:

Порядок совершения операций не важен – каждая цифра изменится на противоположную столько раз, сколько эта цифра упоминается в наборе операций. А значит количество начальных последовательностей зависит только от вариативности получившейся в результате кодирования строки. Количество возможных вариантов строки зависит от числа ? в строке – каждый из них вне зависимости от других может принять значение 0 или 1, таким образом существует 2^n вариантов, где n – число знаков вопроса. В этом варианте задаче получиться могло две штуки - 1 0 1 0 1 1 1 0 или 0 0 1 0 1 1 1

0. В результате расшифровки каждой из этих двух получится какая-то одна и та же последовательность, вне зависимости от порядка действий.

4. Комбинаторика (3 балла)

[Два палиндрома]

Чебурашка в школе недавно прошёл тему «Прогрессии». Больше всего ему понравились арифметические прогрессии, ему стало интересно, сколько существует возрастающих арифметических прогрессий из трёх натуральных чисел, при этом самое большое число в прогрессии не превосходит 100. Помогите Чебурашке найти количество таких прогрессий.

Пример:

Прогрессия {98, 99, 100} подходит под условия, прогрессия {33, 66, 99} – тоже, а прогрессия {99, 100, 101} – нет.

Ответ: 2450

Решение:

Начнём рассматривать прогрессии при малых максимумах из всех чисел.

При максимуме 1 или 2 количество возможных комбинаций – 0.

При 3:

1, 2, 3

Всего – 1

При 4:

1, 2, 3

2, 3, 4

Всего – 2

При 5:

1, 2, 3

2, 3, 4

3, 4, 5

1, 3, 5

Всего – 4

Заметим, что каждый последующий максимум включает в себя все последовательности предыдущего максимума. Также на каждом увеличении максимума количество возможным прогрессий увеличивается на j (изначально j = 0), но каждые два увеличения максимума j также увеличивается на 1. Можно выразить n-ый член через формулу:

$$P(n) = P(n-1) + \left\lfloor \frac{n-1}{2} \right\rfloor$$

при $n = 0$: $P(n = 0) = 0$;

$\left\lfloor \frac{n-1}{2} \right\rfloor$ – деление на 2 с округлением вниз(до целого)

Пример:

$$\left\lfloor \frac{1-1}{2} \right\rfloor = \left\lfloor \frac{0}{2} \right\rfloor = 0$$

$$\left\lfloor \frac{2-1}{2} \right\rfloor = \left\lfloor \frac{1}{2} \right\rfloor = 0$$

$$\left\lfloor \frac{3-1}{2} \right\rfloor = \left\lfloor \frac{2}{2} \right\rfloor = 1$$

$$\left\lfloor \frac{20-1}{2} \right\rfloor = \left\lfloor \frac{19}{2} \right\rfloor = 9$$

Раскрывая эту формулу для $P(n-1), P(n-2) \dots P(n-(n-1))$-ых элементов получим:

$$P(n) = P(0) + \left\lfloor \frac{1-1}{2} \right\rfloor + \left\lfloor \frac{2-1}{2} \right\rfloor + \left\lfloor \frac{3-1}{2} \right\rfloor + \left\lfloor \frac{4-1}{2} \right\rfloor + \cdots + \left\lfloor \frac{n-1}{2} \right\rfloor = P(0) + 0 + 0 + 1 + 1 + 2 + 2 + \cdots + \left\lfloor \frac{n-1}{2} \right\rfloor$$

Докажем верность данной формулы через математическую индукцию:

Предположим, что формула верна для k-го элемента:

$$P(k) = P(0) + 0 + 0 + 1 + 1 + 2 + 2 + \cdots + \left\lfloor \frac{k-1}{2} \right\rfloor$$

Проверим формулу для k+1-го элемента:

$$P(k+1) = P(0) + 0 + 0 + 1 + 1 + 2 + 2 + \cdots + \left\lfloor \frac{k-1}{2} \right\rfloor + \left\lfloor \frac{k}{2} \right\rfloor$$

Заметим, что формула $P(k+1)$ содержит в себе формулу для $P(k)$:

$$P(k+1) = P(k) + \left\lfloor \frac{k}{2} \right\rfloor$$

Сделаем замену k+1 на n:

$$P(n) = P(n-1) + \left\lfloor \frac{n-1}{2} \right\rfloor$$

Нами была получена изначальная формула, а следовательно формула верна.

Тогда используя формулу:

$$P(n) = P(0) + 0 + 0 + 1 + 1 + 2 + 2 + 3 + 3 + \cdots + \left\lfloor \frac{n-1}{2} \right\rfloor$$

Можно воспользоваться формулой арифметической прогрессии. Так как каждый член мы считает дважды, то можно вынести множитель 2 за скобки. Также легко заметить, что всего членов в прогрессии будет $\frac{n}{2}$, так как каждый член мы считаем дважды; а a_n член получается из формулы выше: $\left\lfloor\frac{n-1}{2}\right\rfloor$

Тогда итоговая формула для подсчёта:

$P(n) = \frac{2*(0+a_n)*\frac{n}{2}}{2} = a_n * \frac{n}{2}$

Тогда при n=100:

$P(100) = a_{100} * \frac{100}{2} = a_{100} * 50$

$a_{100} = \left\lfloor\frac{100-1}{2}\right\rfloor = \left\lfloor\frac{99}{2}\right\rfloor = 49$

$P(100) = 49 * 50 = 2450$

5. Основы логики. (3 балла) [Замена операции]

Петя написал на доске логическое выражение, тождественно равное 1, используя только 4 логические операции: отрицание (not), конъюнкция (and), дизъюнкция (or), импликация (->). Потом пришел Вася и, пока никто не видит, заменил одну логическую операцию в выражении на другую. Помогите Пете восстановить свое выражение, зная только, что Вася точно не изменял операции отрицания.

Выражение, которое осталось на доске после Васи:

F = (A or B) -> ((A or D or B or not A) -> D)

В ответ запишите номер операции (считая операции в выражении слева направо с 1, всего в выражении 7 операций), которую заменил Вася, и затем через пробел обозначение логической операции, которая была заменена Васей.

Ответ: 7 or

Решение:

Для начала составим таблицу истинности для имеющегося выражения.

A	B	D	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Чтобы понять, какую операцию заменил Вася, рассмотрим случаи, когда полученное выражение равно 0.

Заметим, что выгоднее сперва рассмотреть случай A=1, B=1, D=0.

F(1, 1, 0) = (1 or 1) -> ((1 or 0 or 1 or 0) -> 0)

1 2 3 4 5 7

Пронумеруем логические операции в порядке их следования (номер 6 соответствует пропущенному отрицанию), и рассмотрим, какие из них мы можем заменить.

На что бы мы не заменили, у нас левая скобка импликации будет равна 1, а правая – 0. Следовательно, мы не получим выражение тождественно равное 1.

Заметим, что если заменить -> на or, то F(1, 1, 0) = 1

3-4) На что бы мы не заменили, у нас первая скобка будет равна 1, а вторая – 0. Следовательно, мы не получим выражение тождественно равное 1.

5) Если заменить пятую операцию на ->, то F(1, 1, 0) = 1.

7) Если заменить седьмую операцию на or, то F(1, 1, 0) = 1.

Теперь рассмотрим случай A=0, B=1, D=0. Заметим, что в этом случае отвергается вариант с заменой пятой операции, так как:

F = (0 or 1) -> ((0 or 0 or 1 -> 1) -> 0) = 0.

Следовательно, остались варианты с заменой 2-ой или и 7-ой операции на or. Рассмотрим последнюю оставшуюся комбинацию A=1, B=0, D=0. В этом случае также подойдут обе указанные замены.

Таким образом, замены 2-ой или 7-ой операций на or, позволяют получить 1 для комбинаций значений переменных, для которых в исходной таблице истинности были значения 0, но теперь необходимо проверить, что они не приведут к появлению нулевых значений для оставшихся комбинаций.

Нетрудно заметить, что замена 2-ой операции на or приведет к ложному значению функции, например, для A=0, B=0, D=0:

F = (0 v 0) v ((0 or 0 v 0 v 1) -> 0) =0, а вот замена 7-ой операции на or даст истинное значение для всех возможных комбинаций значений логических переменных A, B и D, поэтому, это единственно верный вариант замены. Тогда ответ может быть записан как 7 or

6. Алгоритмизация и программирование. Формальный исполнитель (3 балла) [Кусты смородины]

У бабушки Васи на огороде растет очень много кустов смородины, и Вася решил помочь ей со сбором урожая. Для этого он создал робота, который умеет собирать ягоды.

Кусты в бабушкином огороде высажены в одну прямую линию, всего их - 1455 штук, они пронумерованы слева направо – с 1-го до 1455-го. По Васиной задумке, робот перемещается от куста к кусту, каждый раз «прыгая» от текущего куста на n кустов вправо или влево. Для удобства введем два обозначения: **шагом** робота назовем это положительное число n, а **направлением** робота – направление, в котором он движется. Направление может принимать только два значения - «влево» или «вправо». Обе величины непостоянны; правила, по которым они изменяются при движении робота, описаны ниже. Пример использования обозначений: если робот был на 6-ом кусте, а потом «прыгнул» на шаг = 8 вправо, то он оказался на 14-ом кусте.

Робот перемещается с куста на куст и собирает ягоды по следующему алгоритму:

1. **Шаг** робота становится равным "1", **направление** движения - "вправо".

2. Робот находит куст, ближайший к левому краю участка, такой, с которого еще не собрана смородина и собирает с него все ягоды.

3. Робот перемещается на текущее значение **шага** в текущем **направлении** движения и собирает смородину с куста, на котором оказался. Если ягоды с этого куста уже им собраны, то робот сразу переходит к пункту 4 алгоритма.

4. Значение **шага** увеличивается вдвое, значение **направления** меняется на противоположное (если направление было «вправо», то оно меняется на «влево», и наоборот).

5. Шаги 3-4 повторяются, пока робот находится в пределах участка со смородиной, то есть на одном из кустов.

Как только робот выходит за пределы участка, выполнение алгоритма начинается с пункта 1. Робот полностью прекращает работу, как только собраны ягоды со всех кустов в огороде.

Ягоды с куста с каким номером робот соберет самыми последними, если в начале смородина не была собрана ни с одного куста?

Ответ: 1455

Решение:

Заметим, что робот начнет сбор урожая с первого куста (как самого левого такого, с которого еще не собраны ягоды), число 1 - нечетное. Длина шага в начале равна одному, направление – «вправо» (по пункту 2 алгоритма), значит далее он перейдет на второй куст, число 2 – четное. Далее шаг будет увеличиваться в два раза каждый раз, когда робот будет возвращаться к 3-ему пункту алгоритма, а значит независимо от того, как много шагов успеет сделать робот до возвращения к первому пункту алгоритма (иными словами, до того, как выйдет за границы участка), он точно больше не попадет на куст с нечетным номером – поскольку при добавлении четного числа (шага) к четному номеру куста, на котором робот будет находиться, получится снова четный номер куста. Последнее утверждение верно для старта с любого нечетного куста (не только первого). Формально четность всех следующих номеров кустов можно доказать по индукции.

Заметим также, что, так как первое значение шага при очередной итерации всегда равно 1, то робот всегда соберет ягоды как с куста, с которого началась очередная итерация (предположим, что этот куст - нечетный), так и сразу после этого – с ближайшего к нему справа куста (четного). При этом он гарантированно не попадет на куст с нечетным номером, следующий за этим четным кустом – так как далее все шаги – четные. Таким образом можем гарантировать, что, начав с нечетного куста, когда робот выйдет за границы участка и вернется к пункту 1 алгоритма, ближайший к левому краю куст с несобранными ягодами – будет куст со следующим нечетным числом. Значит, так как мы начали с нечетного куста (первый слева куст), все итерации будут начинаться с нечетного куста. Формально: если мы начинаем с нечетного куста с номером n+1, то следующий куст по алгоритму – n+2, и далее все «проходимые» кусты имеют четные номера, а куст с номером n+3 остается нетронутым – самым левым из еще не тронутых – и с него начнется следующая итерация.

Таким образом, итерации будут начинаться последовательно со всех нечетных кустов, и ягоды с 1455-го куста будут собраны только тогда, когда очередная итерация начнется с именно с этого куста, а это гарантирует, что к этому моменту будут собраны ягоды со всех кустов «левее» (т. е. вообще со всех кустов), и этот куст будет последним.

Наглядный пример к решению.

Для наглядности работы алгоритма рассмотрим конкретный пример действий робота. Итерации разделены вертикальной чертой («|»). После каждой черты – алгоритм начинается с первого пункта.

При количестве кустов = 69 (69 – это просто не слишком большое число, подходящее для приведения примера):

| 1 2 | 3 4 2 6 | 5 6 4 8 | 7 8 6 10 2 18 | 9 10 8 12 4 20 | 11 12 10 14 6 22 | 13 14 12 16 8 24 | 15 16 14 18 10 26 | 17 18 16 20 12 28 | 19 20 18 22 14 30 | 21 22 20 24 16 32 | 23 24 22 26 18 34 2 66 | 25 26 24 28 20 36 4 68 | 27 28 26 30 22 38 6 | 29 30 28 32 24 40 8 | 31 32 30 34 26 42 10 | 33 34 32 36 28 44 12 | 35 36 34 38 30 46 14 | 37 38 36 40 32 48 16 | 39 40 38 42 34 50 18 | 41 42 40 44 36 52 20 | 43 44 42 46 38 54 22 | 45 46 44 48 40 56 24 | 47 48 46 50 42 58 26 | 49 50 48 52 44 60 28 | 51 52 50 54 46 62 30 | 53 54 52 56 48 64 32 | 55 56 54 58 50 66 34 | 57 58 56 60 52 68 36 | 59 60 58 62 54 | 61 62 60 64 56 | 63 64 62 66 58 | 65 66 64 68 60 | 67 68 66 | 69

Если каждый раз записывать только номера кустов, на которые робот попадает в первый раз (и собирает при этом ягоды), то получим:

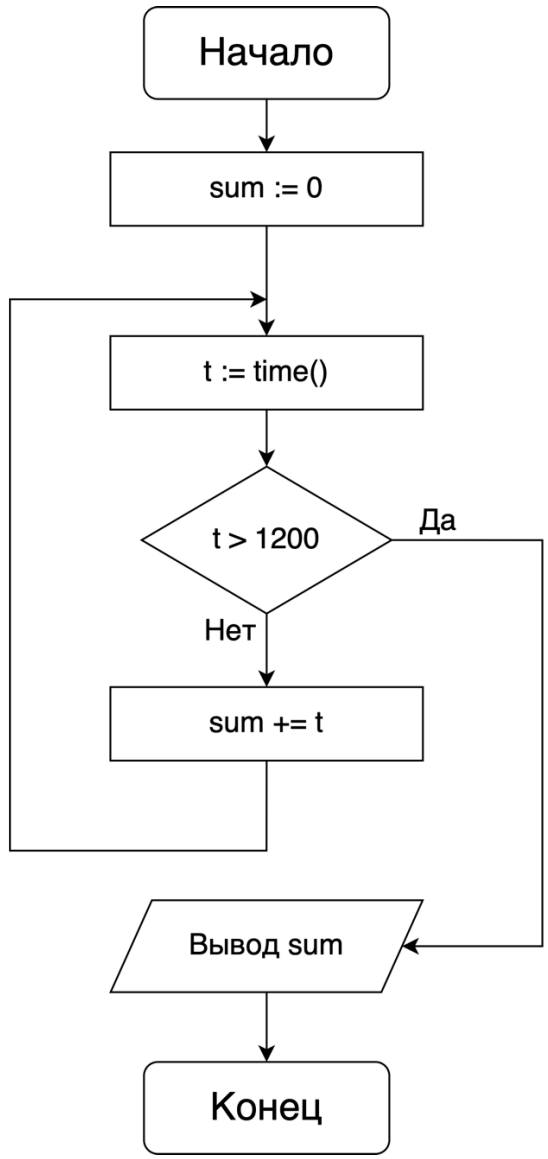
| 1 2 | 3 4 6 | 5 8 | 7 10 18 | 9 12 20 | 11 14 22 | 13 16 24 | 15 26 | 17 28 | 19 30 | 21 32 | 23 34 66 | 25 36 68 | 27 38 | 29 40 | 31 42 | 33 44 | 35 46 | 37 48 | 39 50 | 41 52 | 43 54 | 45 56 | 47 58 | 49 60 | 51 62 | 53 64 | 55 | 57 | 59 | 61 | 63 | 65 | 67 | 69

- Видим, что действительно итерации последовательно начинаются со всех нечетных чисел, и последним кустом, с которого будут собраны ягоды, будет 69-ый.

Прыжки робота по прямой задаются последовательностью действия +1 -2 +4 -8 +16 -32 и т. д.

7. Алгоритмизация и программирование. Анализ блок-схемы (2 балла) [Time()]

Антон недавно узнал, что при написании алгоритмов можно использовать функцию time(). Особенность данной функции заключается в том, что она возвращает количество миллисекунд, прошедших с момента запуска алгоритма. Однако Антону не совсем ясно как анализировать алгоритм с использованием новой функции, поэтому он просит Вас помочь ему с этим.



Какое значение выведет данный алгоритм, если известно, что функция time() и переходы между блоками происходят мгновенно, а также известно время выполнения следующих операций:

Присваивание (оператор :=) — 2 миллисекунды

Инкремент (оператор +=) — 2 миллисекунды

Сравнение (операторы >, <, >=, <=) — 1 миллисекунда

Соответственно в данном алгоритме сначала пройдет 2 миллисекунды на присваивание (sum := 0), далее функция time() вернёт 2, затем на протяжении 2 миллисекунд будет происходить запись этого значения в переменную t и на момент выхода из блока пройдет 4 миллисекунды, а в переменной t будет находится значение 2.

В качестве ответа укажите целое число — значение переменной sum при выводе.

Ответ: 143880

Решение:

Для начала заметим, что sum представляет из себя сумму элементов t, где каждое слагаемое меньше 1200. Проанализируем какие значения принимает переменная t для первых трех шагов:

$$t_1 = 2$$
$$t_2 = 2 + 5 = 7$$
$$t_3 = 2 + 2 * 5 = 12$$

Таким образом последовательность t_i представляет собой арифметическую последовательность, где i-ый элемент имеет вид $t_i = 2 + 5(i - 1)$

Задача сводится к нахождению суммы арифметической прогрессии. Пусть последовательность имеет n элементов. Зная, что последнее слагаемое будет меньше 1200, составим неравенство:

$$2 + 5(n-1) < 1200$$
$$5(n-1) < 1198$$
$$n - 1 < 239.6$$
$$n < 240.6$$

Так как n должно быть целым и максимально возможным, $n = 240$. В таком случае последний элемент суммы примет значение $t_n = 2 + 5(n-1) = 2 + 5(240-1) = 1197$

По формуле суммы арифметической прогрессии:

$$sum = \frac{2 + 1197}{2} \cdot 240 = 143880$$

Соответственное в результате работы будет выведено число 143880

8. Алгоритмизация и программирования. Анализ кода (1 балл) [Еще один шифр]

Вася и Денис хотели общаться друг с другом на неизвестном языке, и Вася написал программу, которая закодировала русский алфавит так, чтобы каждой букве соответствовало некое число. В результате получается целочисленный массив, в котором каждому элементу соответствует буква в порядке следования в алфавите.

Алгоритм:

```
алг кодирование алфавита (арг цел n, рез цел таб ABC[1:n])
нач
    цел n:= 33
    цел таб ABC[1, n]
    нц для i от 1 до n
        ABC[i]:= 1
    кц
    нц для i от 1 до n
        если i mod 2 = 0
           то
               ABC[i]:= i * i
           иначе
               цел j
               нц для j от 1 до i
                   ABC[i]:= ABC[i] * 2
               кц
        все
    кц
кон
```

Маша хотела удивить ребят и тоже использовать их закодированный алфавит.

Вам необходимо помочь Маше закодировать слово АЛГОРИТМ.

a mod b – функция, которая подсчитывает остаток от деления числа a на число b.

В ответе запишите сумму чисел, соответствующих каждой закодированной букве.

Примечание. Русский алфавит: АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Ответ: 9486

Решение:

Изучив псевдокод, можно понять, что:

- букве алфавита с четным номером соответствует число, которое равно номеру буквы во второй степени
- букве алфавита с нечетным номером соответствует число, которое равно значению степени двойки с показателем степени равным номеру буквы в алфавите

Выяснив это, легко посчитать, какое место в алфавите занимает каждая буква слова АЛГОРИТМ, определить закодированное числовое значение каждой буквы и сложить все получившиеся числа. А=2, Л=2^{13}=8192, Г=4^2=16, О=16^2=256, Р=18^2=324, И=10^2=100, Т=20^2=400, М=14^2=196. 2+8192+16+256+324+100+400+196=9486

9. Моделирование на графе (1 балл) [Дополнение до смысла]

Робот Марвин, страдая от скуки, придумал очередной занимательный, но абсолютно бесполезный шифр. Устройство этого шифра выглядит так:

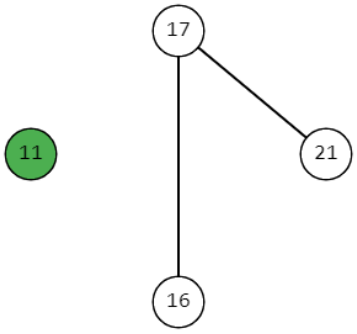
Сообщение передаётся в виде графа, в котором все вершины записаны по окружности. Каждой вершине присвоено какое-то число, не обязательно уникальное, т.е. некоторые вершины могут иметь одинаковые числа. Чтобы расшифровать сообщение, нужно дополнить граф рёбрами так, чтобы для любых двух вершин графа существовало ровно одно ребро, соединяющее их. Затем, для каждой вершины графа нужно умножить число дополненных к этой вершине рёбер на число, находящееся в вершине графа, и взять остаток от деления по модулю на 33. После, следует перевести эти значения в буквы русского языка по таблице и расставить их в порядке движения по окружности по вершинам по часовой стрелке, начиная с вершины, отмеченной зелёным цветом.

0	а	11	к	22	х

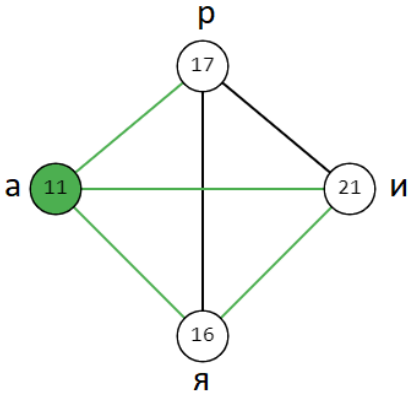
1	б	12	л	23	ц
2	в	13	м	24	ч
3	г	14	н	25	ш
4	д	15	о	26	щ
5	е	16	п	27	ъ
6	ё	17	р	28	ы
7	ж	18	с	29	ь
8	з	19	т	30	э
9	и	20	у	31	ю
10	й	21	ф	32	я

Пример:

Исходный граф:

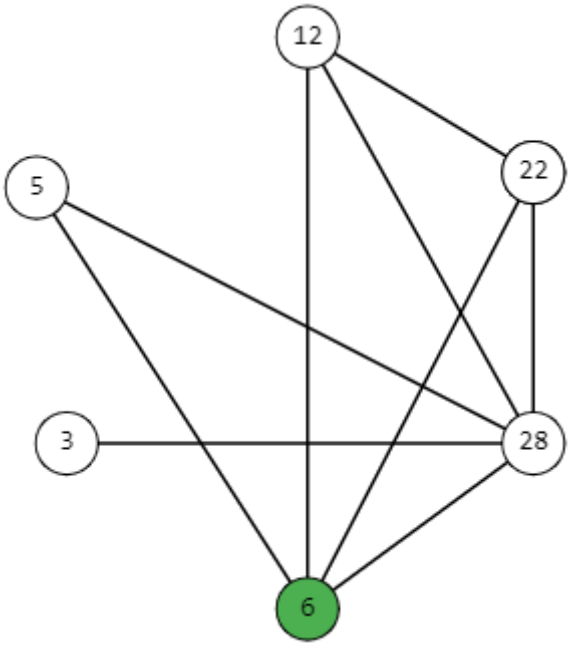


Граф после преобразований:



Ответ для примера: ария

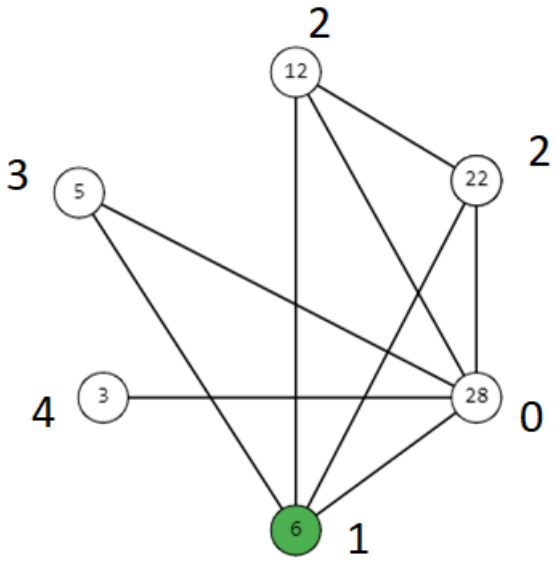
Определите, какое сообщение зашифровано в следующем графе:



Ответ: ёлочка

Решение:

Для того, чтобы для любых двух взятых вершин графа существовало одно ребро, соединяющее их, нужно дополнить граф до полного. У каждой вершины может быть максимально 5 рёбер, так как всего 6 вершин. Посчитаем для каждой вершины значение, равное: 5 - b, где b – число рёбер, выходящих из этой вершины.



Теперь для каждой вершины, умножим значение в вершине на посчитанное число и возьмём остаток по модулю 33. Вершины рассматриваем по часовой стрелке, начиная от зелёной вершины. Получим: 6 12 15 24 11 0 Переведём по таблице в буквы русского алфавита: ё л о ч к а

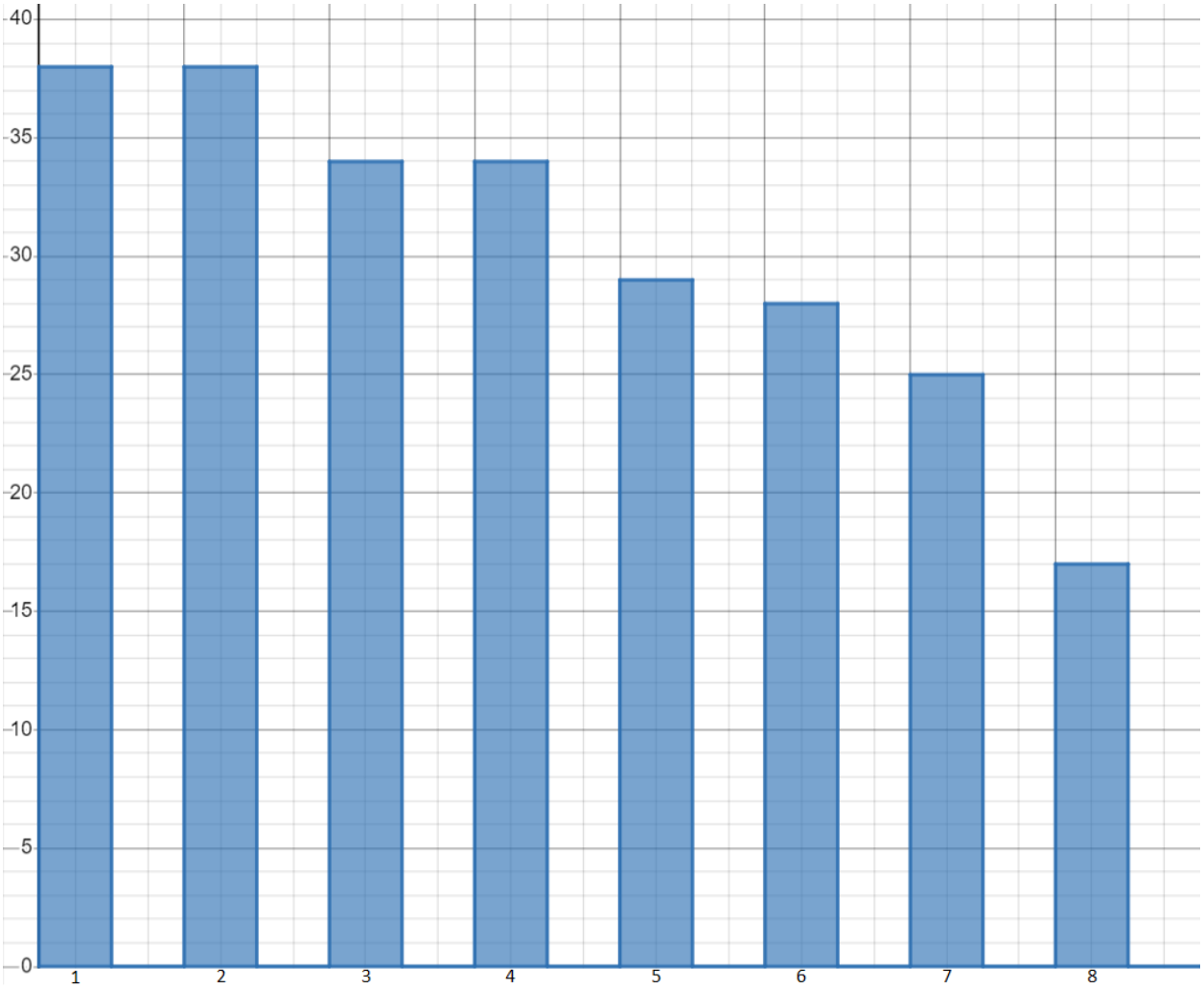
10. Моделирование. Графики и таблицы (2 балла) [Шахматисты]

Проводится большой турнир по различным видам спорта. Последним туром являются шахматы. До турнира по шахматам таблица рейтинга была такой:

Номер участника	Количество баллов
1	36
2	34
3	26
4	30
5	17
6	25
7	29
8	32

Известно, что турнир по шахматам проводился по олимпийской системе (т.е. участники делились по парам, и тот, кто выиграл в каждой паре выходил в следующий этап соревнования). За победу давалось 2 очка. Гарантировалось, что ничьих нет.

Саша, не очень внимательно следящий за соревнованиями, пропустил игру в шахматы. Единственное, что он успел увидеть финальный рейтинг, представленный в виде гистограммы:



Примечание: по оси ординат представлены значения финальных рейтингов каждого участника, значения по оси абсцисс – место участника в финальном рейтинге, а НЕ его номер.

Помогите Саше выяснить, кто занял второе место по шахматам. В ответ запишите номер этого участника. Если существует несколько вариантов таких участников, запишите минимальный номер. Если такого участника не определить, запишите в ответе NULL.

Ответ: 4

Решение:

Максимальное количество баллов, полученное в этом раунде – 6, так как всего 8 участников, следовательно, на первом этапе будет 4 пары, и 4 человека выйдут во второй этап, а во втором этапе будут 2 игры. После будет финал. Таким образом, победитель наберет 6 баллов, игрок, занявший 2 место, наберет 4 балла, еще два игрока наберут по 2 балла и еще 4 игрока наберут по 0 баллов.

Рассмотрим 4 минимальных результата на гистограмме: 17, 25, 28, 29.

17 могло быть получено как 17+0, 15+2, 13+4 или 11+6. В исходной таблице есть только 17, значит, это может быть только игрок 5, и он набрал по шахматам 0 баллов.

25 могло быть получено как 25+0, 23+2, 21+4 или 19+6. В исходной таблице есть только 25, значит, это может быть только игрок 6, и он набрал по шахматам 0 баллов.

28 могло быть получено как 28+0, 26+2, 24+4 или 22+6. В исходной таблице есть только 26, значит, это может быть только игрок 3, и он набрал по шахматам 2 балла.

29 могло быть получено как 29+0, 27+2, 25+4 или 23+6. В исходной таблице есть 25 и 29, но 25 набрал игрок 6, и выше мы доказали, что он мог набрать только 0 баллов. Следовательно, 29 баллов может иметь только игрок 7, и он набрал по шахматам 0 баллов.

Значит, игроки с номерами 5, 6, 9 набрали по шахматам 0 очков, а игрок 3 - 2 очка, и никто из них не занял 2 место.

У игрока 4 было 30 баллов, а набрать он мог или 34 или 38 (все, что осталось на гистограмме). Значит, он набрал 4 балла, и следовательно, занял второе место по шахматам, набрав суммарно 34 балла.

На всякий случай исключим ситуацию несовместных данных. У нас остались неопознанными игроки с номерами 1 (исходно 36 баллов), 2 (34 балла) и 8 (32 балла) и 3 неопознанных суммарных результата (два по 38 и один в 34 балла). Игрок с номером 1 мог набрать только 2 балла (стало 38). Остаются игроки с номерами 2 и 8. Кто-то из них должен набрать 38, и кто-то 34 балла. При этом один может набрать 0, а другой +6. Следовательно, игрок 2 набрал 0 баллов и сохранил свои 34 балла в итоговом рейтинге, а игрок 8 выиграл шахматный турнир, набрал 6 баллов и получил 38 баллов в итоговом рейтинге.

Значит, задача разрешима единственным способом и ответ 4.

Отборочный этап. Первый тур (приведен один из вариантов заданий)

1. Теоретическая информатика, поиск информации (1 балл)

Множественный выбор ответа

[Лицензии]

Однажды Лёва сильно заскучал. Чтобы спастись от скуки он решил найти себе занятие на долгое время. Он решил создать свою операционную систему. Выяснив, что это довольно трудно, он остановился на том, что хочет пока сделать собственный дистрибутив (ядро и набор программ). Но он решил, что программы возьмет, распространяемые под конкретной лицензией (версии лицензий могут отличаться). Помогите Лёве подобрать все программы из предложенных так, чтобы они распространялись под лицензией MPL.

Варианты ответа

1. FileZilla
2. VLC Media Player
3. Mozilla Firefox
4. Apache web server
5. Blender
6. Ни одна из перечисленных

Ответ: 3

2. Кодирование информации. Комбинаторика (2 балла)

[Дружба]

Команда "Дружба" вышла в финал по киданию дротиков, им предстоит сразиться против команды "Львы". В каждой команде по 3 игрока. Каждый игрок может набрать от 1 до 8 очков. Побеждает та, команда, у которой в сумме больше всего очков. Если у обоих команд равное количество очков, то объявляется ничья. В команде «Дружба» Саша набрал 8 очков, а Лена и Максим всего по 1 очку. Ребята очень хотят выиграть и им очень интересно сколько существует вариантов набора очков командой «Львы», при котором команда «Дружба» выигрывает. Помогите им найти это число.

Варианты, отличающиеся игроками, например, вариант, при котором игрок №1 набрал 2 очков, игрок №2 набрал 3 очков, игрок №3 набрал 4 очков, и вариант, при котором игрок №1 набрал 4 очков, игрок №2 набрал 3 очков, игрок №3 набрал 2 очков, отличаются.

В ответе укажите целое число.

Ответ: 84

3. Кодирование информации. Комбинаторика (3 балла)

[Места в поезде]

У Ани запланирован выезд с классом за город на поезде. Так получилось, что образовалось 3 группы из трех и 3 группы из двух человек, таких, что школьники в каждой группе хотят сидеть рядом в одном ряду. Трем ребятам не принципиально с кем сидеть. Для школьников выкуплены билеты в 6 рядах, в каждом ряду по три места. Однако классный руководитель считает, что нельзя удовлетворить желание всех. Аня уверена в обратном. Помогите ей найти число таких рассадок, в которых желания всех школьников будут удовлетворены. Рассадки отличаются, если хотя бы у двоих школьников отличаются места, на которых они сидят. Группы не имеют пересечений. В ответе укажите **сумму цифр** числа всех рассадок.

Ответ: 45

4. Кодирование информации. Комбинаторика, теория игр (2 балла)

[Пульт]

Петя и Вася хотят посмотреть свою любимую передачу «В мире животных» по телевизору, где она идет по 14-му каналу. Но пока до начала осталось немного времени, они решили поиграть в следующую игру:

Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может переключить текущий канал на 2 или на 3 канала вперед (на свой выбор). Например, если сейчас включен пятый канал, то включить можно либо седьмой, либо восьмой канал. Пропускать ход нельзя. Перед началом игры включен первый канал.

Игрок считается победителем в двух ситуациях:

1) Если в свой ход он включил 14-ый канал;

2) Если противник в свой ход попал на канал с номером, большим 14.

Как только один из игроков стал победителем, игра заканчивается.

Кто из игроков может победить в этой игре независимо от ходов противника и какое максимальное количество его ходов ему может на это потребоваться? В ответ запишите через пробел два числа: сначала номер игрока (Петя – 1, Вася – 2, если невозможно сказать - 0), а затем максимальное количество ходов, которое может потребоваться сделать победителю для победы.

Примечания. Считается, что никто из игроков не поддается. Обратите внимание, что спрашивается не общее число ходов, сделанных двумя игроками, а только количество ходов, сделанных одним игроком – победителем.

Ответ: 1 3

5. Кодирование. Комбинаторика (1 балл)

[Охота на зомби]

Программист Василий опытный охотник на зомби. Под названием зомби он, конечно, подразумевает зомби процессы в операционной системе Linux. Это такие процессы, которые уже завершились, но, по каким-то причинам процесс, вызвавший их, не может обработать их коды завершения, и процессы продолжают числиться в таблице всех процессов. Каждый процесс в Linux имеет уникальный номер.

Василий исследует программу с номером 16032. В её коде была допущена ошибка, из-за которой программа стала создавать зомби процессы. Известно, что все зомби процессы, которые созданы этой программой, имеют номера большие, чем её номер. Василий смотрел процессы до номера 52012 и заметил интересную закономерность – у всех номеров зомби процессов модуль разности сумм всех цифр номера на нечётных позициях (позиции в числе номеруются слева направо с единицы) и чётных позициях была равна 21 (например, для номера 12345 такой модуль разности сумм будет равен |(1 + 3 + 5) – (2 + 4)| = 3).

Василию очень интересно узнать, сколько же зомби процессов появились в промежутке между запуском программы и созданием процесса с номером 52012 включительно. В ответ запишите это число.

Ответ: 14

6. Основы логики. (1 балл)

[Стертые переменные]

Саша увлеклась изучением информатики и решала дополнительные задания после урока. Она написала на доске логическую функцию и построила таблицу истинности. Но её хулиган-одноклассник Вова прибежал и стер с доски обозначения переменных в заголовке таблицы.

Помогите Саша восстановить порядок переменных в её таблице истинности для функции.

F(A, B, C) = (не A и C) или (C и B) или (A и не B)

?	?	?	F(A, B, C)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

В ответе укажите последовательность из трех больших латинских букв в порядке их следования в столбцах таблицы истинности.

Ответ: CAB

7. Основы логики. Анализ логических функций (2 балла)

[Три уравнения]

На кружке по информатике ребята разбирали логические уравнение и способы их решения. После занятия лучшие друзья Петя, Ваня и Толя решили закрепить изученный материал. Каждый из них придумал логическое уравнение и решил его. Сверив свои ответы с ответами товарищей, Ваня обратил внимание, что некоторые из их ответов совпадают. Ему стало интересно: из всех ответов, которые получили ребята, сколько получилось уникальных.

Уравнение Пети: $(A \vee C) \wedge B \wedge (\neg A \vee C) \vee C = 0$

Уравнение Вани: $(A \wedge B \vee C) \rightarrow (A \wedge C) = 1$

Уравнение Толи: $(\neg A \vee C) \leftrightarrow (\neg B \vee C \vee A) = 0$

Решением уравнения считается уникальная комбинация значений логических переменных A, B и C, для которой равенство будет истинным. Естественно, одно уравнение может иметь более одного решения.

В ответ укажите число – количество уникальных ответов в трех уравнениях, то есть таких комбинаций логических переменных A, B и C, которые являются решением хотя бы одного из перечисленных уравнений.

Примечание:

Используемым логическим операциям соответствуют следующие таблицы истинности:

A	B	¬ A	A ∧ B	A ∨ B	A → B	A ↔ B
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Ответ: 6

8. Основы логики. Кодирование (3 балла)

[Побитовое шифрование]

Для алфавита из 8 букв дана таблица кодирования букв в виде трехразрядных двоичных кодов:

М	К	А	Р	О	Л	Е	П
000	001	010	011	100	101	110	111

Для некоторого слова, используя двоичные коды его букв, составили новую двоичную последовательность следующим образом. Двигаясь слева направо, рассматривается очередная пара соседних букв в слове и для неё записывается результат одной из трех побитовых операций над двоичными кодами этих букв:

- если обе буквы согласные: записывается операция побитового И

- если обе буквы гласные: записывается операция побитового ИЛИ
- если буквы разной гласности: записывается побитовый XOR

Обратим внимание, что коды всех букв, кроме первой и последней буду участвовать в двух побитовых операциях.

Например, для слова МАК результатом кодирования будет последовательность 010011.

Расшифруйте последовательность 011110000010 и запишите в ответ получившееся слово.

Ответ: ЛЕММА

9. Системы счисления (1 балл) [Смена оснований]

Никита очень любит системы счисления. Он решил складывать числа в разных системах счисления. Сначала он взял числа 1700_8, 170_8, 17_8, посчитал их сумму и перевел полученное число в двоичную систему счисления. Затем Никита решил посчитать количество единиц в полученном двоичном числе и взять его как основание новой системы счисления n. В этой системе счисления он сложил числа 1400_n, 140_n, 14_n. Сколько одинаковых цифр используются в записи суммы 1700_8, 170_8, 17_8 в восьмеричной системе счисления и в записи суммы 1400_n, 140_n, 14_n в системе счисления с основанием n соответственно?

В ответ укажите число - количество уникальных совпадающих цифр в двух записях чисел.

Например:

В числах 1478_9 и 7651_8 совпадают две цифры - 1 и 7

В числах 123_8 и 2256_7 совпадает только одна цифра - 2

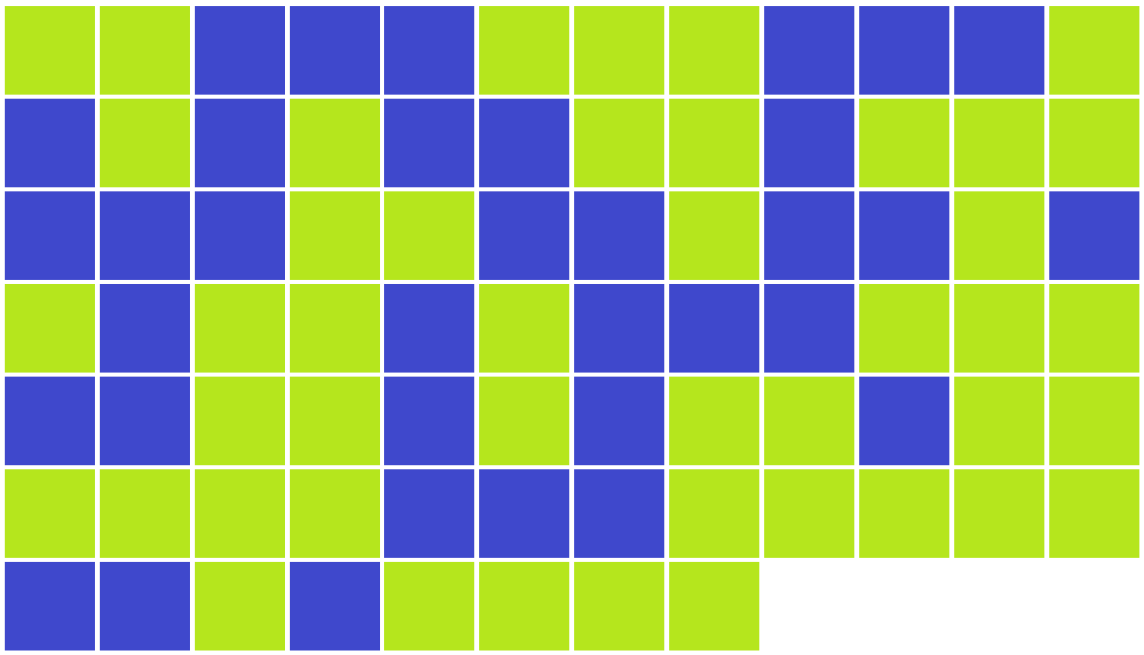
Ответ: 3

10. Системы счисления (2 балла) [Биты чётности]

В алфавите 116 символов, пронумерованных от 0 до 115. Каждый номер переводится в двоичную систему счисления и представляется одним байтом (допускаются незначащие нули в записи отдельного номера). Каждый байт разбивается на две части по 4 бита и перед каждыми 4 битами ставится один дополнительный бит - бит чётности. Бит чётности равен 0, если в следующих четырех битах четное количество единиц и равен 1, если нечетное. Таким образом, кодом каждого символа является последовательность из 10 бит. Затем, все коды записываются друг за другом в общую последовательность бит, и эта последовательность записывается слева направо сверху вниз в виде изображения, где бит равный нулю отмечается зелёным квадратом, а бит равный единице – синим. Если последовательность не помещается на одну строку, то она переносится на следующую. Для большей надёжности сразу после окончания записи общей последовательности бит помещается еще одна её копия.

Программист Арсений решил передать своему другу сообщение, закодированное подобным образом, но канал передачи оказался недостаточно защищён от внешнего воздействия и из-за помех некоторые цвета квадратов поменялись на противоположные. Гарантируется, что на каждые 5 бит, начинающихся с бита чётности придётся не более одной помехи и только в одной из копий записи общей последовательности.

Нужно расшифровать сообщение и вывести через пробел последовательность номеров, записав каждый из них в десятичной системе счисления.



Ответ: 115 92 24 100

Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Теоретические основы информатики (1 балл)

[Аппаратные характеристики]

Архитектура компьютера состоит из нескольких основных компонентов. Характеристикой каких компонентов является тактовая частота?

1. Процессор
2. Оперативная память
3. Блок питания
4. Жёсткий диск
5. Материнская плата

Ответ: 1,2

2. Алгоритмизация. Моделирование по строке (2 балла)

[Очень длинные строки]

Даны 2 строки:
X = “uklamut”
Y = “aklea”

К данным строкам применен следующий алгоритм:
1) Строка X переворачивается
2) В конец строки X дописывается строка Y
3) Строка Y переворачивается

Данный алгоритм повторяется k раз. Индексация символов в строке начинается с нуля.

Найдите такое минимальное k, что после выполнения алгоритма k раз в строке X символом с индексом 99 будет символ ‘a’.

В ответ укажите искомое значение k.

Ответ: 40

3. Алгоритмизация. Анализ кода (1 балл)

[Матрица]

Учитель информатики дал Маше псевдокод алгоритма по заполнению ячеек матрицы. Маша выполнила алгоритм и получила матрицу, заполненную числами.

```
алг заполнение матрицы числами (арг цел N, рез таб A)
нач
    ввод N
    цел таб A[0:N-1, 0:N-1]
    нц для i от 0 до N-1
        нц для j от 0 до N-1
            если i = j
               то
                  A[i, j]:= i + j
            все
            если i < j
               то
                  A[i, j]:= j - i
            все
            если i > j
               то
                  A[i, j]:= i - j
            все
        кц
    кц
    вывод A
кон
```

После выполнения алгоритма Маше стало интересно, какой будет сумма чисел на побочной диагонали матрицы. Помоги Маше вычислить эту сумму.

На вход алгоритма подается N = 101 – число строк и столбцов матрицы. Нумерация элементов матрицы начинается с [0, 0]. При обращении к элементам матрицы первая координата отвечает за номер строки, вторая – номер столбца.

Главной диагональю матрицы называется диагональ, проведенная из левого верхнего угла матрицы в правый нижний.

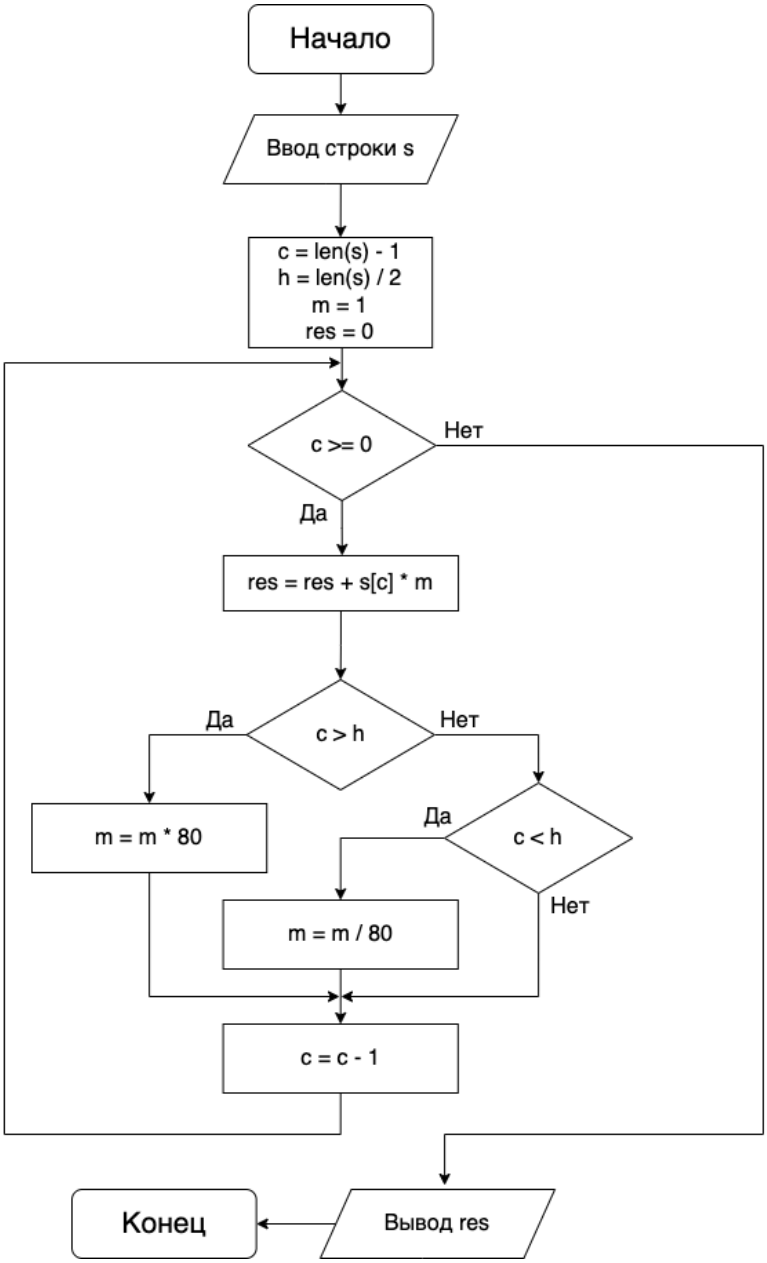
Побочной диагональю матрицы называется диагональ, проведенная из левого нижнего угла матрицы в правый верхний.

В ответе укажите одно число – сумму чисел на побочной диагонали матрицы.

Ответ: 5200

4. Алгоритмизация. Анализ блок-схемы (3 балла)
[Хэш-функция]

Даня придумал свою хэш-функцию (функция, которая ставит в соответствие строке числовое значение), и ему стало интересно как много существует слов длины 6, которые образуют коллизию со словом “ЗАДАЧА”. Коллизией называется ситуация, когда хэш-функция при разных входных строках возвращает одно и то же значение. В ответе укажите целое число – количество слов, длины 6, которые образуют коллизию с указанным словом.



Примечания:

Алфавит состоит из заглавных букв русского алфавита (33 буквы), где каждой букве соответствует код, равный её позиции в алфавите (А – 1; Б – 2; В – 3; …).

Функция len(s) возвращает длину строки s

s[i] возвращает код i-ой буквы строки s; нумерация по i начинается с 0.

Ответ: 1124

5. Алгоритмизация. Анализ блок-схемы (3 балла)
[Преобразование строк]

Робот Марвин был вынужден сотни лет сидеть без дела, умирая от скуки и, чтобы придать своему существованию какой-то смысл придумал абсолютно бессмысленный алгоритм. Алгоритм получает на вход строку текста, делает определённые преобразования и возвращает строку.

Также в алгоритме используется функция number(), которая для переданной буквы вернёт целое число – её номер из таблицы:

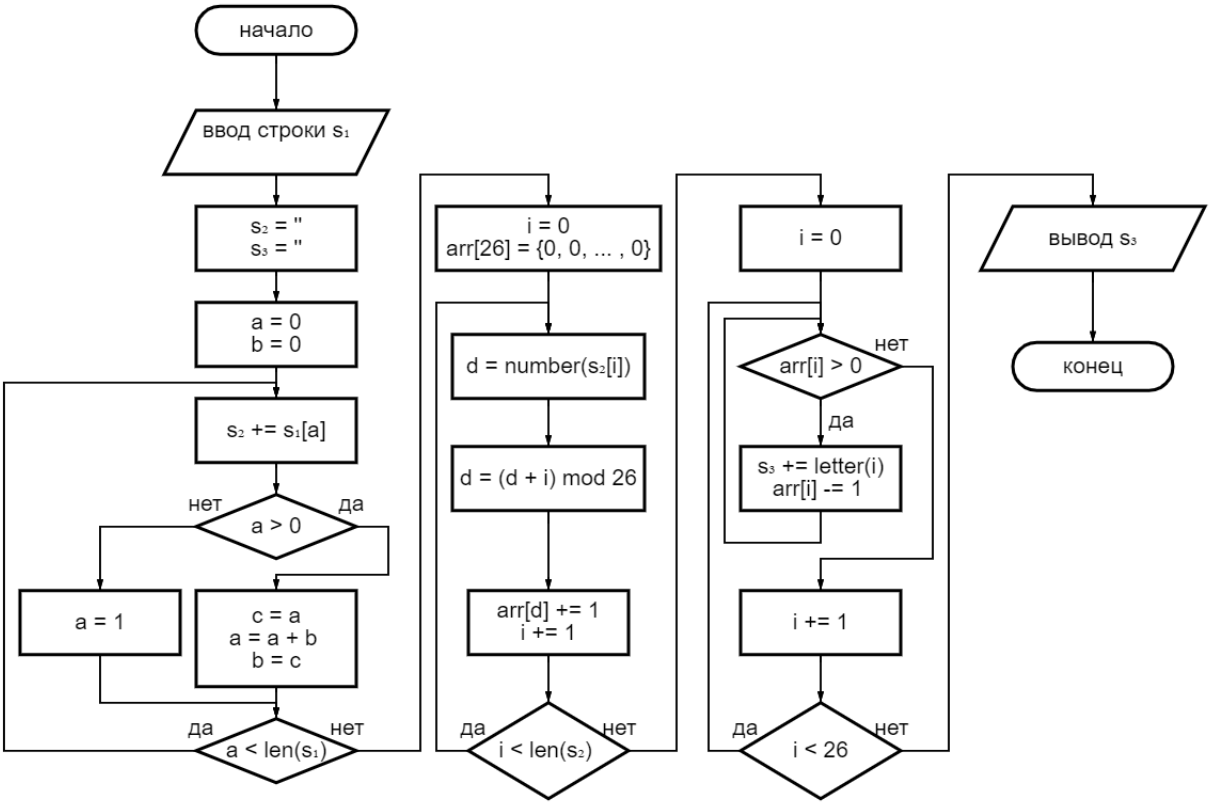
a = 0	g = 6	m = 12	s = 18	y = 24
b = 1	h = 7	n = 13	t = 19	z = 25

c = 2	i = 8	o = 14	u = 20
d = 3	j = 9	p = 15	v = 21
e = 4	k = 10	q = 16	w = 22
f = 5	l = 11	r = 17	x = 23

Ответ не обязан иметь смысл и быть каким-то существующим словом.

Определите, что будет выведено для строки:

cantlieinfrontofthebulldozer



Примечания:

letter() – функция получает номер и возвращает букву по таблице.

s_1, s_2, s_3 – строки. Прибавление элемента к строке означает добавление символа в конец строки.

s_2 = '', s_3 = '' – инициализация переменных s_2, s_3 пустыми строками

a += b ↔ a = a + b; a -= b ↔ a = a - b

arr[n] = {0, 0, … , 0} – массив из n элементов, заполненный нулями.

a mod b – остаток от целочисленного деления числа a на число b.

len(s) – функция возвращает число элементов в строке.

Ответ: abccnqttx

6. Сортировка и фильтрация данных (2 балла)

[Книжная полка]

Ваня решил навести порядок в книжном шкафу. У него есть полное собрание детской энциклопедии – 16 томов, каждый том имеет свой номер. Сейчас они расставлены на полке шкафа в следующем порядке:

5 3 13 10 11 16 6 14 4 12 2 1 7 15 8 9

Ваня хочет расставить их по возрастанию номера тома. При этом он не хочет снимать с полки больше одной книги за раз. Перестановку книг он выполняет по следующему алгоритму:

1. Выбрать одну книгу и снять ее с полки

2. Вставить вынутую книгу между двумя другими книгами на этой же полке (книги легко двигаются по полке влево-вправо, между двумя книгами можно последовательно вставить сколько угодно книг)

При необходимости повторить шаги 1-2.

При этом во время уборки Ваня хочет снимать с полки наименьшее возможное число книг. Какое наименьшее возможное число книг ему придется вынуть и потом поставить на полку в другое место, чтобы расставить книги по порядку?

Пример:

Если нужно правильно расставить 4 книги, изначально расположенные в следующем порядке: 1 4 3 2, Ваня может привести полку в порядок, переставив две книги, например:

1) вынуть второй том и поставить его между первым и четвертым (получится 1 2 4 3)

2) вынуть третий том и поставить его между вторым и четвертым (получится 1 2 3 4). В ответ в данном случае записывается «2» - так как с полки снималось две книги.

Ответ: 11

7. Сортировка и фильтрация данных (1 балл)
[Просроченные заказы]

В базе данных пункта выдачи заказов на момент 15.09.2022 имеется следующая информация:

Таблица 1:
Первый столбец – номер заказа
Второй столбец – дата привоза заказа в пункт выдачи

Номер заказа	Дата привоза на пункт выдачи
1-123	03.09.2022
3-133	03.09.2022
1-125	06.09.2022
1-134	04.09.2022
6-128	10.09.2022
2-123	01.09.2022

Таблица 2:
Первый столбец – номер заказа
Второй столбец – стоимость заказа
Третий столбец – сколько дней заказ может храниться бесплатно в пункте выдачи

Номер заказа	Стоимость	Срок хранения
1-1xx	345	7
2-1xx	410	5
1-1xx	345	6
3-1xx	833	2
6-1xx	3934	7
1-1xx	675	11
2-1xx	221	11
2-1xx	600	4
3-1xx	2546	7
1-1xx	234	7
1-1xx	67	5

В таблице 2 могут содержаться записи о заказах, которые еще не доставлены в пункт выдачи, но о каждом заказе, указанном в таблице 1 есть запись в таблице 2. При этом, отображение таблицы настроено таким образом, что скрывает последние две цифры номера заказа, заменяя их на xx.

Известно, что за если товар лежит в пункте выдачи дольше своего срока хранения, то за каждый день превышения срока хранения покупателю за свой заказ придется заплатить 10% от стоимости этого товара. Нужно определить минимальную сумму, которую могут заплатить покупатели за превышение срока хранения. В ответе запишите число с точностью до одного знака после запятой.

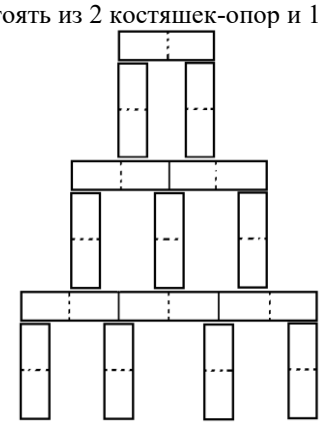
Ответ: 993 || 993,0

8. Моделирование (2 балла)
[Домино]

Костя решил построить небоскреб из костяшек домино.

Примечание: Один набор домино включает в себя 28 костяшек, на каждой из которых по два числа – оба от 0 до 6. В одном полном наборе каждая возможная пара чисел встречается ровно 1 раз (дубли, т. е. пары типа 1,1 или 0,0 также присутствуют).

Структура небоскреба из домино строго определена: здание строится этажами снизу вверх и слева направо; этаж ниже текущего имеет ровно на 1 костяшку-опору и на 1 костяшку-крышу больше, чем текущий; верхний этаж обязательно должен состоять из 2 костяшек-опор и 1 костяшки крыши. На рисунке для примера показана структура трехэтажного небоскреба.



Назовем сумму очков на костяшке *ценностью* этой костяшки. Перед началом строительства все костяшки раскладываются на столе по группам по значению их ценности. Выбор каждой следующей костяшки из еще не задействованных в строительстве костяшек (для последующей установки ее на ее место в здании) всегда производится по следующим правилам:

6. Если существует группа, в которой костяшек больше, чем в любой другой группе, то берется любая костяшка из этой группы.
7. Если же групп с наибольшим количеством костяшек (по сравнению с другими группами) на данный момент несколько, то берется любая костяшка из группы с наименьшей ценностью из этих групп.

Пример логики выбора костяшки:

Если в какой-то момент времени на столе лежит 3 костяшки с ценностью = 6 и 4 костяшки с ценностью = 7, то для продолжения строительства выбирается любая из костяшек с ценностью 7 (по первому из правил выбора костяшки). После того, как эта костяшка помещена на свое место в небоскребе, на столе остается по 3 костяшки с ценностями 6 и 7, и следующей на своё место в здании отправляется любая костяшка с ценностью 6 (по второму из правил выбора костяшки).

У Кости есть 5 полных наборов домино. Костя хочет построить небоскреб с максимальным возможным числом законченных этажей (часть костяшек может остаться незадействованными в строительстве, если с помощью них нельзя построить еще один законченный этаж).

Какова будет ценность костяшки, которая окажется самой верхней? В ответе укажите целое число.

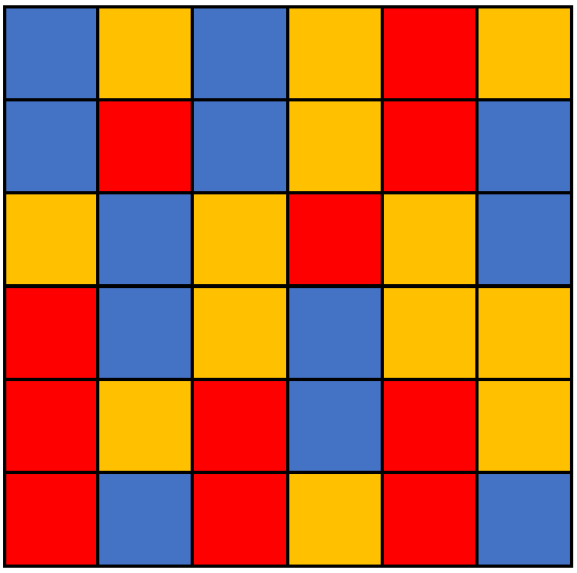
Ответ: 5

9. Моделирование (2 балла)
[Разноцветная жизнь]

Дан квадрат 6x6, каждая клетка которого раскрашена в определенный цвет. За один ход все цвета клеток могут поменяться по следующим правилам:

8. если клетка граничит (по стороне или по углу) хотя бы с тремя клетками такого же цвета, то эта клетка становится черной
9. если клетка граничит (по стороне или по углу) хотя бы с тремя черными клетками, то эта клетка становится черной

Нужно определить, можно ли превратить данный квадрат в «черный квадрат» за не более чем 10 ходов. И если квадрат не может стать черным за не более чем 10 ходов, то нужно определить, какое минимальное количество клеток нужно перекрасить (в один из имеющихся на квадрате цвет), чтобы можно было сделать квадрат черным не более чем за 10 ходов.



В ответ запишите через пробел два числа: за какое число ходов исходный квадрат можно сделать черным (-1 если это невозможно за 10 или менее ходов) и какое минимальное количество клеток нужно перекрасить (если черный квадрат достигается из исходного за 10 или менее шагов, то в ответ записать 0).

Ответ: 10 0

10. Моделирование (1 балл)
[Кубики]

Ульяниной младшей сестре Полине на Новый год подарили большую коробку с кубиками. Девочкам очень понравилось строить из этих кубиков башенки.

Во время очередной игры Ульяна вспомнила, как на уроке информатики они проходили простые числа, и ей стало интересно, сколько можно построить башенок из кубиков, если количество кубиков в каждой башне будет простым числом.

Изначально все кубики находятся в коробке. Для первой башни Ульяна дала сестре 2 кубика, для каждой последующей - следующее простое число кубиков. Количество кубиков в каждой башне должно быть уникальным.

Полина же на недавнем уроке математики изучала деление и решила, что если сумма цифр числа делится на 5, то такое число ей не нравится. В этом случае башенку она строить не будет, кубики, которые дала Ульяна, возвращаются в коробку. Игра заканчивается, когда оставшихся в коробке кубиков меньше, чем очередное простое число.

В коробке всего 150 кубиков. Вопрос: какое максимальное количество башенок может построить Полина?

В ответ укажите одно число - количество башенок.

Примечание:

Простое число - число больше 1, делителями которого являются только 1 и само число.

Пример: число 17 - простое, так как оно делится только на 1 и на 17.

Число 32 не нравится Полине, так как сумма цифр числа 32 равна 5, а это число делится на 5.

Ответ: 8