Задания для 10 класса

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

1. Кодирование информации. Системы счисления (1 балл) [XYZ_2024]

Дано выражение:

$$XYZ_9 + ZXY_{12} + YZX_{14} = 2024_{10}$$

В данном выражении X, Y и Z – допустимые различные цифры указанных систем счисления. Определите значения переменных Y и Z, если X = 5. В ответе укажите через пробел 2 целых положительных числа: сначала значение Y и затем значение Z.

Ответ: 6 2

Решение:

Так как Y и Z – допустимые цифры девятеричной системы счисления, то $0 \le Y, Z \le 8$

Переведём все числа данного выражения в десятичную систему:

$$XYZ_9 = 5 * 9 * 9 + Y * 9 + Z = 405 + 9Y + Z$$
$$ZXY_{12} = Z * 12 * 12 + 12 * 5 + Y = 144Z + 60 + Y$$
$$YZX_{14} = Y * 14 * 14 + Z * 14 + 5 = 196Y + 14Z + 5$$

Просуммируем слагаемые левой части:

$$405 + 9Y + Z + 144Z + 60 + Y + 196Y + 14Z + 5 = 206Y + 159Z + 470$$

Приравняем левую и правую часть, упростим:

$$206Y + 159Z + 470 = 2024$$
$$206Y + 159Z = 1554$$

Теперь необходимо более точно определить границы допустимых значений для Y и Z.

Во-первых,

$$1554 = 206Y + 159Z \ge 159(Y + Z)$$

Значит, $(Y + Z) \le 9$.

Во-вторых,

$$1554 = 206Y + 159Z \le 206(Y + Z)$$

Значит, $(Y + Z) \ge 8$.

В-третьих, так как число 1554 чётное, то либо оба слагаемых слева нечётные, либо оба чётные. Так как число 206Y не может быть нечётным вне зависимости от значения Y, то оба слагаемых чётные. Число 159Z чётное только при чётных значениях Z. Значит, возможные значения для числа 159Z – это 0, 318, 636, 954 и 1272. Соответствующие этим значениям значения для 206Y – 1554, 1236, 918, 600, 282. Заметим, что число 206Y делится нацело на 206. Из перечисленных 5 чисел таким свойством обладает только число 1236. Значит, Y=1236/206=6, Z=2. Ответ: 6 2.

Также возможно и решение задачи программированием, например, так:

```
from itertools import product
for y, z in product(range(9), repeat=2):
    num = int(f'5{y}{z}', 9) + int(f'{z}5{y}', 12) + int(f'{y}{z}5', 14)
    if y != z and num == 2024:
        print(y, z)
```

2. Кодирование информации. Объем информации (1 балл) [Цифровой диктофон]

Друг Пети, Павел, пытается сконструировать цифровой диктофон и просит Петю написать прошивку для кодирования и сохранения в памяти оцифрованного аудиосигнала. Петя решил, что будет записывать данные без сжатия и оцифровывать аудиосигнал с частотой дискретизации 88200 Hz, выбрав такую глубину кодирования, чтобы в каждый отсчет времени сохранялось одно из возможных 65536 значений сигнала (для записи значения сигнала в каждый отсчет времени Петя использует минимальное, одинаковое для всех возможных значений количество бит). Поскольку Петя предполагает использовать пару микрофонов, Павел решил записывать звук двухканальным, сохраняя оцифрованный аудиосигнал с указанными параметрами независимо для каждого канала. Опытный Вася обратил внимание Пети на две возможности для уменьшения памяти. Во-первых, можно уменьшить частоту дискретизации в два раза, а во-вторых, записывать с выбранной глубиной кодирования только один канал, а для второго канала записывать для каждого отсчета времени только разность значения сигнала со значением в первом канале, считая, что для этого хватит 1024 возможных значений (для записи разности сигналов в каждый отсчет времени предлагается также использовать минимальное, одинаковое для всех возможных значений разности количество бит). Петя принял оба предложения Васи и обнаружил, что для аудиосигнала длительностью t секунд удалось сэкономить больше 20 МБайт памяти. Определите минимальное **целое** значение t, при котором это возможно. В ответе укажите целое число.

Примечания:

1. При записи оцифрованного сигнала в память не записывается никакая дополнительная информация.
2. 1 МБайт=2^{20} байт.

Ответ: 101

Решение:

Запишем формулу для определения информационного объема в первоначальном формате записи:

$2 * t * 88200 * \log_2(65536) = t * 88200 * 2 * 16$ бит.

После изменения формата, частота дискретизации составит 88200/2=44100 Hz, а глубина кодирования для второго канала станет равна $\log_2(1024) = 10$ бит. Следовательно, формула для определения информационного объема будет: $t * 44100 * 16 + t * 44100 * 10 = t * 44100 * 26$ бит.

Определим значение t, при котором экономия памяти составила бы ровно 20 МБайт:

$t * 88200 * 2 * 16 - t * 44100 * 26 = 12 * 1024 * 1024 * 8$

$t = 20 * 1024 * 1024 * 8 / (88200 * 2 * 16 – 44100 * 26) = 100{,}11$ секунд

Следовательно, если бы длительность аудиосигнала была 100 секунд, разность составила бы меньше 20 МБайт, значит ответ – 101 секунда.

3. Основы логики (1 балл)

[Четыре импликации]

Два набора значений переменных A, B и C называют не эквивалентными, если значение хотя бы одной переменной различается.

Сколько существует не эквивалентных друг другу наборов значений переменных, при которых равенство выполняется при любом значении D:

$$(((D \to (A \wedge B \wedge C)) \to (A \wedge \mathrm{B} \vee \mathrm{C})) \to (\mathrm{A} \vee \mathrm{B} \wedge \mathrm{C})) \to (A \vee \mathrm{B} \vee \mathrm{C}) = 1$$

В ответ укажите одно целое число.

Решение:

В первую очередь составим таблицу истинности для используемых выражений:

A	B	C	A∧B∧C	A∧B∨C	A∨B∧C	A∨B∨C
0	0	0	FALSE	FALSE	FALSE	FALSE
0	0	1	FALSE	TRUE	FALSE	TRUE
0	1	0	FALSE	FALSE	FALSE	TRUE
0	1	1	FALSE	TRUE	TRUE	TRUE
1	0	0	FALSE	FALSE	TRUE	TRUE
1	0	1	FALSE	TRUE	TRUE	TRUE
1	1	0	FALSE	TRUE	TRUE	TRUE
1	1	1	TRUE	TRUE	TRUE	TRUE

Рассмотрим таблицу истинности для импликации:

A	B	A->B
0	0	1
0	1	1
1	0	0
1	1	1

Заметим, что при B=1 результат импликации не зависит от A и равен 1. То есть, равенство $(\ldots) \to (A \vee \mathrm{B} \vee \mathrm{C}) = 1$ всегда верно при $(A \vee \mathrm{B} \vee \mathrm{C}) = 1$. Значит, 7 из 8 наборов значений A, B и C удовлетворяют условию. Проверим оставшийся набор. При A=0, B=0, C=0 выражение примет вид: $(((D \to 0) \to 0) \to 0) \to 0$. Рассмотрим значения этого выражения при D=0 и D=1

D	$D \to 0$	$(D \to 0) \to 0$	$((D \to 0) \to 0) \to 0$	$(((D \to 0) \to 0) \to 0) \to 0$
0	1	0	1	0
1	0	1	0	1

Таким образом, результат меняется в зависимости от D, что противоречит условию. То есть, 8-й набор не подходит, и ответ – 7.

4. Кодирование информации. Формальные исполнители (2 балла)

[6 букв]

Есть исходная строка S, состоящая из 6 символов. Результирующая строка R изначально пустая и формируется следующим образом: N раз выполняются следующие два шага:

3. Дописать в конец строки R строку S.
4. Циклически сдвинуть строку S на один символ влево.

Например, для строки S=’abcdef’ и N=4 получится строка R=’abcdefbcdefacdefabdefabc’.

Для некоторой строки S получили результирующую строку R для $N=10^{10}$.

Известны некоторые символы в строке R (нумерация символов строки начинается с 1 слева направо):

Номер символа	Значение
10^8+2	a
10^8+4	c
10^8+8	b
10^8+16	d
10^8+3	f
10^8+5	e

Определите и введите в ответ строку S, для которой это возможно.

Если вариантов таких строк несколько, введите любую подходящую.

Если такой строки не существует, введите в ответ NULL.

Ответ: cedabf

Решение:

Обратим внимание, что, поскольку в исходной строке 6 символов, возможно сделать последовательно 6 циклических сдвигов, после чего опять получится исходная строка. Следовательно, в результирующей строке будут повторяться одинаковые последовательности из 36 символов.

Построим такую последовательность (вручную или с помощью простой программы), взяв за основу строку «123456»:

123456234561345612456123561234612345

Возьмем первый номер из таблицы: 10^8+2 и поделим на 36 с остатком. Остаток будет равен 30. Следовательно, это 30-ый символ в этой строке – символ «4». Следовательно, четвертый символ в исходной строке, символ “a”.

Аналогично поступим с остальными номерами символов и получим следующую таблицу:

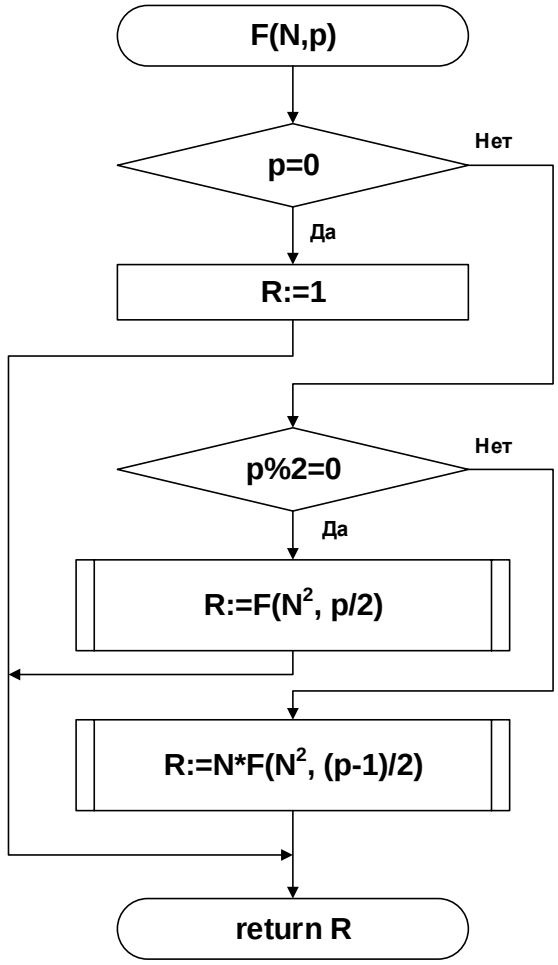
Номер символа в результирующей строке	Значение	Номер символа в исходной строке
10^8+2	a	4
10^8+4	c	1
10^8+8	b	5
10^8+16	d	3
10^8+3	f	6
10^8+5	e	2

Обратим внимание, что мы определили все 6 символов исходной строки. Запишем их в правильном порядке и получим ответ: cedabf.

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

[Рекурсия]

Дана блоксхема алгоритма, реализованного в виде рекурсивно вызываемой функции:



Найдите такую пару целых положительных чисел N и p (известно, что p>1), чтобы вызов F(N,p) вернул число 387420489. Если таких пар существует несколько, найдите ту, у которой максимальное значение N. В ответе укажите через пробел сначала значение N и затем значение p. Если такой пары не существует, укажите в ответе NULL.

Ответ: 19683 2

Решение:

Проанализировав алгоритм, представленный в виде блок-схемы, можно понять, что он реализует возведение числа N в степень p. Следовательно, нужно подобрать такие N и p, чтобы $N^p = 387420489$.

Для этого разложим данное в условие число на сомножители, например, с помощью такого программного решения:

```
ans=[]
d=2
while Z>1:
    if Z%d==0:
        ans.append(d)
        Z//=d
    else:
        d+=1
print(ans)
```

Результатом будет список: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3], то есть 18 чисел 3. Значит $387420489 = 3^{18}$.

Но в условии сказано, что необходимо найти пару с максимальным значением N.

Заметим, что 18 = 9*2. Тогда $3^{18} = (3^9)^2 = 19\,683^2$. И это пара с максимальным значением N, поскольку значение p – минимально возможное (по условию p>1). Следовательно, ответ: 19683 2.

6. Телекоммуникационные технологии (3 балла). [Обрыв канала]

В ОС семейства GNU/Linux существует возможность объединения сетевых интерфейсов (сетевых карт) в группы (*bonding*). Ее используют для балансировки нагрузки при передаче и обеспечения отказоустойчивости.

В результате объединения в системе создается виртуальный сетевой интерфейс `bond`. С точки зрения остальных слоев сетевого стека этот интерфейс – обычная сетевая карта, однако при передаче данных через нее реальная передача данных осуществляется через тот или иной физический сетевой интерфейс по выбору ядра операционной системы. Выбор физического интерфейса зависит от режима *bonding*.

Существует 7 режимов *bonding*. Один из них – `balance-xor`. Когда физические интерфейсы объединяются в этом режиме, для выбора физического интерфейса, через который следует передать кадр канального уровня, используются

значения адресов канального уровня (MAC-адресов), содержащиеся в заголовке кадра. Расчет ведется по следующей формуле:

`(MAC_отправителя XOR MAC_получателя) % число_физических_интерфейсов = индекс_физического_интерфейса`

Здесь `XOR` – побитовая операция, а `%` - операция получения остатка от деления. Значения индекса физического интерфейса для передачи начинаются с нуля.

В случае недоступности части физических интерфейсов передача идёт только по активным интерфейсам (то есть изменяется число физических интерфейсов), а индексы самих интерфейсов пересчитываются, сохраняя изначальный порядок в конфигурации и нумерацию с нуля.

Рассмотрим систему с настроенным виртуальным сетевым интерфейсом `bond` в режиме `balance-xor`, который включает в себя 4 физических сетевых интерфейса с индексами от 0 до 3. Этой системе поступает несколько кадров для передачи. MAC-адрес отправителя равен `24:01:C7:A3:8B:7E`, а MAC-адреса получателей указаны ниже.

```
00:24:1E:6D:FC:7D
44:45:53:36:A0:88
78:84:3C:94:06:A3
00:22:4C:FA:BE:C9
44:45:53:DC:E6:7A
68:76:4F:9A:99:FF
E0:0C:7F:53:C5:09
44:45:53:7C:70:65
BC:6E:64:3C:AD:EF
A4:5C:27:35:CE:7C
44:45:53:75:DA:1F
9C:5C:F9:74:17:50
00:09:BF:10:AD:FD
44:45:53:74:2A:F4
00:EB:2D:38:D0:F2
00:27:09:00:AC:1B
44:45:53:DC:53:CE
68:76:4F:18:EC:5C
64:B5:C6:58:97:DB
44:45:53:60:96:FE
00:1E:45:9C:9C:FF
E8:DA:20:CF:BB:CB
44:45:53:18:37:68
4C:21:D0:35:31:22
00:1F:C5:A3:8B:43
44:45:53:0B:25:65
00:21:9E:91:53:6B
00:26:59:B4:AE:9A
44:45:53:76:95:14
30:17:C8:8A:84:15
```

После передачи определённого количества кадров произошёл обрыв сети, из-за чего одновременно стали недоступны два физических интерфейса из четырёх, и оставшиеся кадры передавались по одному из двух активных интерфейсов. Известно, что по интерфейсу с исходным индексом 0 было передано 7 кадров, по интерфейсу с исходным индексом 1 – 12 кадров, по интерфейсу с исходным индексом 2 – 6 кадров, по интерфейсу с исходным индексом 3 – 5 кадров. Определите, какие из интерфейсов стали недоступны и сколько кадров было передано до обрыва канала. В ответе укажите три числа через пробел: номера интерфейсов (0, 1, 2 или 3) в порядке возрастания и количество переданных кадров. Если есть несколько вариантов того, какие интерфейсы могли стать недоступны, выберите любой вариант. Если есть несколько вариантов того, сколько кадров могло быть передано до обрыва, выберите максимальное число.

Ответ: 2 3 25

Решение:

Для решения задачи нам нужно посчитать побитовое исключающее ИЛИ для 30 пар 48-битных чисел и потом взять остаток от деления на количество активных интерфейсов. Можно заметить, что активных физических интерфейсов в любой момент времени либо 4, либо 2. Это позволяет нам посчитать исключающее ИЛИ только для последних 2 бит чисел пары и потом брать остаток от деления. Вычислим выражение из условия для каждой пары:

Адрес	Индекс интерфейса
`00:24:1E:6D:FC:7D`	3
`44:45:53:36:A0:88`	2
`78:84:3C:94:06:A3`	1

00:22:4C:FA:BE:C9	3
44:45:53:DC:E6:7A	0
68:76:4F:9A:99:FF	1
E0:0C:7F:53:C5:09	3
44:45:53:7C:70:65	3
BC:6E:64:3C:AD:EF	1
A4:5C:27:35:CE:7C	2
44:45:53:75:DA:1F	1
9C:5C:F9:74:17:50	2
00:09:BF:10:AD:FD	3
44:45:53:74:2A:F4	2
00:EB:2D:38:D0:F2	0
00:27:09:00:AC:1B	1
44:45:53:DC:53:CE	0
68:76:4F:18:EC:5C	2
64:B5:C6:58:97:DB	1
44:45:53:60:96:FE	0
00:1E:45:9C:9C:FF	1
E8:DA:20:CF:BB:CB	1
44:45:53:18:37:68	2
4C:21:D0:35:31:22	0
00:1F:C5:A3:8B:43	1
44:45:53:0B:25:65	3
00:21:9E:91:53:6B	1
00:26:59:B4:AE:9A	0
44:45:53:76:95:14	2
30:17:C8:8A:84:15	3

Дальше для рассмотрения возьмём вариант 1.

Нетрудно заметить, что в случае исправности всех интерфейсов в течение всего времени работы получилось бы следующее распределение кадров по интерфейсам:

Индекс	Количество (теор.)	Количество (факт.)
0	6	7
1	10	12
2	7	6
3	7	5

Видно, что интерфейсы с индексами 0 и 1 обработали больше кадров, а с индексами 2 и 3 - меньше, соответственно, первые два числа в ответе - 2 и 3.

Осталось понять, после какого кадра произошёл обрыв, для этого нужно посмотреть, когда мы передадим 1 лишний кадр через интерфейс 0 и 2 лишних кадра через интерфейс 1. Начнём пересчитывать номера интерфейсов ещё раз, но с конца:

Адрес	Индекс исходного интерфейса до обрыва	Индекс исходного интерфейса после обрыва	Не на тот интерфейс?
…	…	…	…
4C:21:D0:35:31:22	0	0	Нет
00:1F:C5:A3:8B:43	1	1	Нет
44:45:53:0B:25:65	3	1	Да
00:21:9E:91:53:6B	1	1	Нет
00:26:59:B4:AE:9A	0	0	Нет
44:45:53:76:95:14	2	0	Да
30:17:C8:8A:84:15	3	1	Да

Видно, что до обрыва должно быть передано максимум 25 кадров, чтобы получилась описанная в условии ситуация.

7. Технологии обработки информации в электронных таблицах, технологии сортировки и фильтрации данных (2 балла)

[Остаться самим собой]

Дана таблица в режиме отображения формул:

	A	B	C	D	E	F	G
1		2	3	4	5	6	
2	2	=IF(MOD(A1; POW(B$1; $A2))=0; DIVIDE($A$1; POW(B$1; $A2)); MOD($A$1; POW(B$1; $A2)))					
3	3						
4	4						
5	5						
6	6						
7	7						
8	8						

Формулу из ячейки B2 скопировали во все ячейки диапазона B2:F8. После чего оказалось, что число из ячейки A1 встречается в диапазоне B2:F8 ровно 4 раза. Определите минимальное и максимальное возможное число в ячейке A1, при котором это могло произойти. В ответ запишите сначала меньшее из значений, а затем большее из них.

Примечание: таблица соответствия имён используемых функций.

	Google Sheets	**Excel**	**LibreOffice**
Условное вычисление	IF	ЕСЛИ	IF
Возведение в степень	POW	СТЕПЕНЬ	POWER
Остаток от деления	MOD	ОСТАТ	MOD
Частное от целочисленного деления	DIVIDE	ЧАСТНОЕ	QUOTIENT

Ответ: 65536 78124

Решение:

Обозначим как X величину POW(B$1; $A2). Формула из ячейки B2 записывает в ячейку либо остаток от деления числа из ячейки A1 на X, если число из ячейки A1 делится не на X и результат деления в обратном случае. Следовательно, число из ячейки A1 (для краткости далее будем называть его просто A1) может попасть в одну из ячеек из диапазона B2:F8 двумя способами: либо если A1 делится на X и при этом A1/X=A1, т.е. X = 1, что не выполняется в данной таблице. Либо A1 не делится на X и тогда в ячейке будет значение A1 mod X. Данная величина будет равна A1 только в том случае, если A1 меньше X. То есть, A1 встретится в диапазоне B2:F8 четыре раза, если ровно 4 возможных значения POW(B$1; $A2) будут больше A1. Для каждого числа из диапазона B2:F8 определим значение X:

	A	B	C	D	E	F
1		2	3	4	5	6
2	2	4	9	16	25	36
3	3	8	27	64	125	216
4	4	16	81	256	625	1296
5	5	32	243	1024	3125	7776
6	6	64	729	4096	15625	46656
7	7	128	2187	16384	78125	279936
8	8	256	6561	65536	390625	1679616

Если A1 будет больше либо равно 78125, то только 3 числа в таблице будут больше A1. Значит, максимально возможное значение – 78124. Если A1 будет 65535 или меньше, то 5 чисел таблицы будут больше A1. Значит, A1 должно быть хотя бы 65536. Значит, ответ на данную задачу 65536 78124.

8. Технологии программирования (3 балла)

[Окна]

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

На экране монитора размером $w \times h$ расположены n прямоугольных окон нового текстового редактора. Размер экрана измеряется в символах, то есть текстовая строка длины w символов занимает всю ширину экрана от левого края до правого, а всего на экране могут поместиться друг под другом ровно h таких строк. Занумеруем строки экрана от 1 до h, а столбцы — от 1 до w, и будем обозначать координатами (i, j) позицию символа на пересечении i-й строки и j-го столбца.

Каждое из n окон содержит только текстовое поле, которое по ширине и высоте помещает в себе целое число символов. Окна могут быть разного размера и расположены так, что каждая из $w \times h$ позиций символов на экране принадлежит текстовому полю ровно одного окна. Иными словами, окна не могут накладываться, но могут касаться границами, и в совокупности покрывают всю площадь экрана.

Изначально все текстовые поля в окнах пустые, то есть не содержат никакой текст. Команды вывода строки на экран задаются тройками вида (r, c, s), означающими, что надо поставить курсор на c-ю слева позицию в r-ю сверху строку

монитора и, начиная с этой позиции, напечатать строку s. Символы строки печатаются по очереди слева направо, каждый следующий символ занимает следующую позицию в той же строке и записывается в ней вместо того, что стояло на этой позиции ранее. Так происходит, пока курсор не дойдет до правой границы текущего окна, после чего поведение курсора зависит от *режима*, в котором работает то окно, в котором он находится.

Окна могут работать в трех режимах. При достижении курсором правой границы окна:

В режиме «clip» вывод останавливается, и оставшаяся часть строки просто не выводится.

В режиме «wrap» курсор переносится на первую (ближайшую к левой границе окна) позицию **этого же окна** в следующей строке, после чего вывод продолжается. Если же текущая строка является последней строкой в текущем окне, вывод останавливается;

В режиме «overflow», если сейчас курсор находится в позиции экрана (r, c), он переносится в позицию $(r, c + 1)$, то есть в следующую позицию на экране, оставаясь в той же строке и игнорируя правую границу окна; после чего вывод продолжается. Если же текущая позиция является последней позицией на экране, то есть $c = w$, курсор переносится на первую позицию на экране в следующей строке $(r + 1,1)$. Если и строка была последняя, то есть курсор достиг правой-нижней позиции на экране ($r = h$ и $c = w$), вывод останавливается.

Для каждого окна известно, в каком режиме оно изначально работает. Обработайте команды вывода строк и команды изменения режимов окон и выведите состояние экрана после обработки всех команд.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$).

Каждый набор входных данных начинается со строки, в которой даны три целых числа n, w и h — количество окон на экране и размеры экрана, соответственно ($1 \le w, h$; $1 \le n \le w \cdot h \le 2000$).

Следующие n строк набора входных данных описывают окна; i-я строка описывает i-е окно и содержит через пробел четыре целых числа $r_{i,1}$, $c_{i,1}$, $r_{i,2}$ и $c_{i,2}$ — номер строки и столбца левого-верхнего угла окна и номер строки и столбца правого-нижнего угла окна ($1 \le r_{i,1} \le r_{i,2} \le h$; $1 \le c_{i,1} \le c_{i,2} \le w$), а также строку mode_i, равную «clip», «wrap» или «overflow» — изначальный режим, в котором работает i-е окно. Гарантируется, что весь экран покрыт окнами и что каждая позиция на экране принадлежит ровно одному окну.

Затем в отдельной строке следует целое число q — количество команд, которые вам предстоит обработать ($1 \le q \le 2000$).

В i-й из следующих q строк дано описание i-й команды в формате

«print r_i c_i s_i», если это команда вывода строки s_i, начиная с позиции (r_i, c_i) ($1 \le r_i \le h$; $1 \le c_i \le w$; $1 \le |s_i|$);

«mode t_i m_i», если это команда изменения режима окна номер t_i на m_i ($1 \le t_i \le n$; $m_i \in$ «clip», «wrap», «overflow»).

Гарантируется, что все s_i состоят из маленьких букв латинского алфавита (символы от ‘a’ до ‘z’), и что сумма длин s_i по всем командам в рамках одного набора входных данных не превосходит $10\,000$.

Формат выходных данных

Для каждого набора входных данных выведите состояние экрана после обработки всех q команд. Состояние экрана — это h строк по w символов, каждый из которых равен букве на соответствующей позиции экрана, либо ‘.’, если соответствующая позиция пустая и не содержит символ.

Пример

Стандартный ввод	**Стандартный вывод**
2 2 2 2 1 1 2 1 clip 1 2 2 2 overflow 4 print 1 1 abcd print 1 2 xxxx mode 1 wrap print 1 2 yyyy 4 3 3 1 1 1 1 overflow 1 2 1 3 clip 2 1 3 1 wrap 2 2 3 3 wrap 8 print 1 1 abcd mode 2 overflow print 1 2 efgh mode 3 overflow	ay y. aef ijy zt.

Стандартный ввод	Стандартный вывод
print 2 1 ijkl print 2 3 x mode 4 overflow print 2 3 yzt	

Замечание

Для удобства восприятия наборы входных данных и соответствующий им вывод в примерах в условии отделяются друг от друга пустыми строками. В реальных тестах этих **пустых строк нет**!

Решение:

Это задача на реализацию. Требовалось внимательно прочитать условие и реализовать то, что в нем просили, добавив минимальные необходимые оптимизации.

Заведем

- text[r][c] – символ, стоящий на позиции (r,c);
- id[r][c] – номер окна, покрывающего позицию с координатами (r,c);
- mode[i] – режим, в котором работает окно номер i;
- left[i] и bottom[i] – номер левого столбца и нижней строки i-го окна, соответственно.

Изначально необходимо просто заполнить таблицу id в соответствии с данными в условии, после чего приступить к обработке запросов. Для каждого запроса будем поддерживать текущее положе- ние курсора (r, c) и очередной символ строки, который надо вывести. Обозначим также за $\mathrm{id}_{\mathrm{cur}}$ номер текущего окна, то есть id[r][c].

Тогда сначала присвоим в text[r][c] текущий символ, а затем переместим курсор.

- Если $c < w$ и id[r][$c+1$] = $\mathrm{id}_{\mathrm{cur}}$, то есть мы еще не дошли до правой границы окна, просто увеличим c на 1. Иначе, посмотрим на mode[$\mathrm{id}_{\mathrm{cur}}$] - режим текущего окна.
- Если режим равен «clip», сделаем break из цикла, отвечающего за вывод, чтобы его остано- вить.
- Если режим равен «wrap», перенесем курсор в $r \leftarrow r+1$ и $c \leftarrow \mathrm{left}[\mathrm{id}_{\mathrm{cur}}]$. Следует отдельно проверить, что если $r = \mathrm{bottom}[\mathrm{id}_{\mathrm{cur}}]$, то тоже надо сделать break.
- Если же окно работает в режиме «overflow», то либо курсор переместится в $(r,c+1)$, либо в
- $(r+1,1)$, либо просто следует сделать break, если достигнут правый нижний угол экрана.

Каждый запрос в таком случае обрабатывается за $O(|s_i|)$, и в сумме все решение работает за $O(wh + \sum |s_i|)$. Если же каждый раз искать номер текущего окна перебором, не храня id[r][c], решение с большой вероятностью не будет проходить по времени.

В другом варианте этой задачи режимам «clip» и «overflow» соответствовали «fill» и «break», а «clip» был заменен на «repeat». В режиме «repeat» достаточно было просто перемещать курсор на $(r, \mathrm{left}[\mathrm{id}_{\mathrm{cur}}])$.

9. Технологии программирования (3 балла)

[Скрещивание]

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

Вы — владелец зоопарка, в котором сейчас обитают n редких экзотических видов змей, i-й из которых имеет *опасность* a_i.

Держать животных в неволе вам не очень хочется, но и выпускать опасных змей в общее пространство тоже, разумеется, нельзя. Для исправления этой проблемы вы решили скрестить некоторые виды, от чего их экзотичность меньше не станет, а вот опасность может уменьшиться. Для одного скрещивания вы можете выбрать два вида змей под номерами i и j, и скрестить их в новый вид, имеющий опасность $a^*_{i,j} = a_i \oplus a_j$, где за $\oplus$ обозначена операция побитового исключающего «ИЛИ» (также обозначается как xor или ^).

Чтобы не получать слишком похожие виды, каждый из имеющихся n видов может участвовать только в одном скрещивании. Также невозможно скрестить два вида, хотя бы один из которых уже является результатов скрещивания каких-то из изначальных n видов. Иными словами, скрещивать можно только исходные виды, и только по парам.

В конце вы получаете новый набор видов, в который войдут исходные виды, не поучаствовавшие в скрещиваниях, а также все результаты скрещиваний. То есть исходные виды, которые были скрещены с какими-то еще, в этот набор не войдут (опять же, потому что нет смысла отводить в зоопарке отдельное место под несколько близких видов — посетители все равно не увидят разницу).

Определите, какие пары видов змей следует скрестить, чтобы максимум из опасностей полученного набора видов был как можно меньше.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$). Далее следуют сами наборы входных данных.

В первой строке набора входных дано целое число n — количество видов экзотических змей в наличии изначально ($1 \le n \le 50\,000$). Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^5.

Во второй строке перечислены n целых чисел a_i — значения опасности этих видов ($0 \le a_i \le 10^9$).

Формат выходных данных

Для каждого набора входных данных выведите в отдельной строке единственное целое число — минимальное возможное значение максимальной опасности, которое можно получить такими скрещиваниями.

Пример

Стандартный ввод	**Стандартный вывод**
4 3 1 2 3 7 1 4 9 16 25 36 49 6 0 1 2 5 6 7 10 5 7 14 10 12 2 1 13 3 11	1 21 4 8

Замечание

В первом примере выгодно скрестить 2 и 3, и получить исходный вид с $a_1 = 1$ и новый с $a_{2,3} = 1$.

Во втором примере скрещиваются 16 с 25 (получается $a_{4,5} = 9$) и 36 с 49 (получается $a_{6,7} = 21$).

В третьем примере скрещиваются 2 с 6 и 5 с 7.

Решение:

Пойдем с конца: пусть ответом на задачу является число x. Посмотрим на его битовую запись и найдем в ней первую (старшую) единицу lead(x). Пусть она стоит на позиции y с конца, то есть соответствует 2^y при разложении x в сумму степеней двойки. Тогда заметим, что любой a_i, у которого lead(a_i) > y, то есть старшая единица стоит раньше, должен быть поставлен в пару с a_j, у которого все биты старше y совпадают с a_i, чтобы их xor давал в старших битах 0.

Таким образом, чтобы в ответе lead был равен y, все a_i с lead(a_i) > y должны разбиваться на пары с одинаковым $\text{head}_{y+1}(a_i) = a_i >> (y+1)$. Здесь за >> обозначен битовый сдвиг вправо, то есть операция, отбрасывающая последние биты числа. В данном случае $a_i >> (y+1)$ отбрасывает все биты с нулевого по y-й включительно.

Алгоритм получается следующий: переберем y от 29 до 0 (можно вместо перебора сделать двоичный поиск, чтобы немного увеличить эффективность), сгруппируем все a_i по их head_y, то есть по $30 - y$ старшим битам, и проверим, что все группы, у которых head_y ненулевой, содержат четное число элементов. Если это так, то можно добиться того, чтобы во всех $30 - y$ старших битах у каждого числа в ответе стояли нули, просто сгруппировав исходные числа по парам внутри групп.

Как только мы нашли такой y, для которого хотя бы одна из групп оказалась нечетного размера, понятно, что в ответе в соответствующем бите будет стоять 1. При этом по всем head_{y-1} группы все еще четного размера, поэтому числа надо разбивать на пары внутри таких групп. Тогда сгруппируем a_i по их head_{y-1} и внутри каждой группы выделим те, у которых следующий бит равен 0 и те, у которых следующий бит равен 1.

После этого для каждой группы решим задачу независимо.

Если количество тех, у которых следующий бит равен 1, четно, то их можно внутри этой группы разбить на пары так, чтобы все результирующие числа имели в следующем бите 0, и в таком случае они все будут меньше ответа.

Иначе можно разбить на пары все, кроме одного. А еще одно число либо оставить как есть, и тогда ответ для этой группы просто будет равен минимальному из чисел с единицей в сле- дующем бите, либо сопоставить ему в пару число из той же группы, но с нулем в следующем бите.

В таком случае достаточно найти $\min_{\substack{a_i \in group0 \\ a_j \in group1}} a_i \oplus a_j$,

где group0 и group1 - элементы текущей группы с нулем или единицей в следующем бите, соответственно.

Поиск такого минимума является классической задачей, которая решается с помощью битового бора: будем воспринимать каждое число как строку из 30 нулей и единиц, добавим все элементы одной подгруппы в такой бор, а для каждого элемента второй подгруппы будем искать ему пару, минимизирующую их xor, стараясь каждый раз в боре идти по совпадающему биту.

Теперь, когда в каждой группе получено минимально возможное максимальное число, ответом будет максимум из ответов для всех групп. Общее время работы такого решения - O(n log A), где A - ограничение сверху на значения a_i.

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

1. Кодирование информации. Системы счисления (1 балл)

[XYZ_2024]

Дано выражение:

$$XYZ_9 + ZXY_{12} + YZX_{14} = 2024_{10}$$