Муниципальный этап всероссийской олимпиады школьников по информатике
Москва, 16 декабря 2018 г.
Разбор заданий для 7–8 классов

Задача 1. Римские числа

Один древнеримский торговец брал несколько раз ссуду в древнеримском банке. Каждый раз банкир записывал размер выданной ссуды на листе пергамента, используя римские числа. Но ввиду дороговизны пергамента, запись производилась плотно и все числа оказались записанными подряд, без разделителей. Когда торговец пришёл возвращать ссуду, оказалось, что невозможно установить разбиение записи на отдельные числа.

Например, если на пергаменте записана строка «XIIV», её можно разбить на римские числа разными способами, например, XI + IV = 11 + 4 = 15 или XII + V = 12 + 5 = 17, возможны и другие варианты разбиения.

Торговец хочет вернуть как можно меньше денег, поэтому он хочет так разбить строку цифр на корректные римские числа так, чтобы сумма всех чисел была как можно меньше.

Вам необходимо решить задачу для следующих пяти строк.

VXIVVIX
CXLVLXC
XIXCXIXL
XCMLXCD
VLMIXCD

Запишите в ответе пять чисел (арабскими цифрами в десятичной системе счисления), каждое число в отдельной строке — ответы для данных пяти строк точно в таком же порядке. Например, для строки «XIIV» ответом будет число 15. Если вы не можете дать ответ для какой-либо строки, напишите в ответе для этой строки число 0.

Правила записи римских чисел

Римскими цифрами можно записать целые числа от 1 до 3999. Число представляется в виде суммы тысяч, сотен, десятков и единиц. Далее из следующей таблицы берётся по одному элементу, соответствующему тысячам, сотням, десяткам, единицам ровно в таком порядке.

Цифра	Тысячи	Сотни	Десятки	Единицы
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Если число тысяч, сотен, десятков, единиц равно 0, то из соответствующего столбца ничего не берётся. Например, число 1990 записывается, как 1000 + 900 + 90 = MCMXC.

Решение

Заметим, что в большинстве случаев неважно, как строка разбивается на слагаемые, так как каждая цифра обозначает некоторое число само по себе, и значения цифр складываются. Например, запись XVI будет равна 10 + 5 + 1 = 16 независимо от того, как разбивать её на слагаемые.

Особенности появятся, только при записи цифр 4, 9, 40, 90, 400, 900, то есть в случаях, когда происходит «вычитание»: запись IX обозначает 9, а на 1 + 10 = 11.

Поэтому на первые два задания легко дать ответ.

VXIVVIX = V + X + IV + V + IX = 5 + 10 + 4 + 5 + 9 = 33

CXLVLXC = C + XL + V + L + XC = 100 + 40 + 5 + 50 + 90 = 285

В задании 3 появляются «двойные» вычитания: строки вида IXL и IXC. Их можно интерпретировать по-разному, например, IXL = I + XL = 1 + 40 = 41 или IXL = IX + L = 9 + 50 = 59. Для уменьшения суммы их нужно разбивать так: IXL = I + XL, IXC = I + XC. Ответ на задание 3: XIXCXIXL = X + I + XC + X + I + XL = 10 + 1 + 90 + 10 + 1 + 40 = 152.

Аналогично, в задании 4 подстроки XCD и XCM нужно представлять в виде XCD = X + CD, XCM = X + CM. Ответ на задание 4: XCMLXCD = X + CM + L + X + CD = 10 + 900 + 50 + 10 + 400 = 1370.

Наконец, в задании 5 появлются «Тройные» вычитания: IXCD = IX + CD. Ответ на задание 5: VLMIXCD = V + L + M + IX + CD = 5 + 50 + 1000 + 9 + 400 = 1464.

Задача 2. Номера домов

На школьном этапе олимпиады была задача по программированию, в которой нужно было посчитать расстояние между двумя домами на улице. В этой задаче человек идёт по улице и складывает номера домов, находящиеся с одной стороны улицы (то есть все номера домов имеют одинаковую чётность). Он начал с дома номер a и закончил домом номер b. Определите сумму всех номеров домов, которые он сложил (он складывал все числа одной чётности начиная с числа a до числа b включительно, числа a и b одной чётности, $a \le b$). Например, если $a = 3$ и $b = 7$, то он сложил числа 3 + 5 + 7 и получил в ответе 15.

Ответом на эту задачу является некоторое выражение, которое может содержать целые числа, переменные a и b (записываемые английскими буквами), операции сложения (обозначаются «+»), вычитания (обозначаются «−»), умножения (обозначаются «*»), деления (обозначаются «/») и круглые скобки для изменения порядка действий. Запись вида «2a» для обозначения произведения числа 2 и переменной a неверная, нужно писать «2 * a».

Если ваше решение работает только в одном случае: для чётных или нечётных чисел a и b, вы всё равно получите неполный балл за эту задачу.

Пример правильной формы записи ответа.

$$b / 2 + (a * b - b) * 2$$

Решение

Воспользуемся методом Гаусса — разобьём все числа на пары с равной суммой: a и b, a + 2 и b - 2 и т. д. В одной паре сумма чисел равна a + b. Общее количество чисел в ряду равно (b - a) / 2 + 1, значит, количество пар будет равно ((b - a) / 2 + 1) / 2. Перемножим два числа, получим ответ:

(a + b) * ((b - a) / 2 + 1) / 2

Ответ можно записать в виде любого выражения, эквивалентного данному.

Задача 3. Сортировка

На уроке информатики учитель показывает фокус с сортировкой чисел. На столе стоят 4 шкатулки, обозначенные слева направо буквами A, B, C, D. Ученик кладёт в каждую шкатулку лист бумаги, на котором написано число от 1 до 4, все числа различные. Учитель не знает, какое число лежит в каждой шкатулке.

Затем учитель пишет на листе бумаги последовательность команд и отдаёт лист ученику. Каждая команда содержит в себе две разные буквы. Ученик последовательно выполняет эти команды. Он открывает две шкатулки, указанные в команде, достаёт из них листки с числами, сравнивает эти числа, и затем листок с меньшим из чисел кладёт в левую открытую шкатулку (шкатулку с меньшей буквой), а листок с большим из чисел— в правую открытую шкатулку (шкатулку с большим номером).

Например, пусть в шкатулках лежат числа 2, 1, 4, 3.

Шкатулка	A	B	C	D
Число внутри неё	2	1	4	3

Учитель написал на листке следующие команды.

AB
BC

Ученик открывает шкатулки A и B, достаёт из них числа 2 и 1. В шкатулку A кладёт число 1, в шкатулку B — число 2.

Шкатулка	A	B	C	D
Число внутри неё	1	2	4	3

Затем ученик открывает шкатулки B и C, достаёт из них числа 2 и 4. В шкатулку B кладёт число 2, в шкатулку C — число 4. Содержимое шкатулок не изменилось.

Фокус заключается в том, что в результате исполнения алгоритма числа в шкатулках оказываются упорядочены, независимо от того, какие числа первоначально лежали в шкатулках. То есть в шкатулке A оказывается число 1, в шкатулке B — число 2, в шкатулке C — число 3, в шкатулке D — число 4.

Помогите выполнить такой фокус — придумайте такую последовательность команд, которая упорядочивает числа в шкатулках при любом первоначальном расположении чисел.

Ответ на эту задачу нужно записать в виде нескольких строк. Каждая строка содержит две различные буквы из множества A, B, C, D. Чем короче будет ваша последовательность команд, тем больше баллов вы получите.

Решение

Самое короткое решение состоит из пяти команд.

AB
CD
AC
BD
BC

Пояснение. После выполнения команды AB в шкатулке A будет лежать наименьшее из чисел в шкатулках A и B, а в шкатулке B — наибольшее из этих чисел. Аналогично после выполнения команды C в шкатулке C будет наименьшее из чисел C и D, а в шкатулке D — наибольшее. Теперь после выполнения команды AC в шкатулке A будет наименьшее из всех чисел, а после выполнения команды BD в шкатулке D будет наибольшее из всех чисел. Наконец, командой BC упорядочиваются два оставшихся числа.

Задача 4. Двоичное кодирование

При двоичном кодировании буквам сопоставляются последовательности из символов «0» или «1». Например, пусть дан следующий код:

Код буквы «К»: 0
Код буквы «Н»: 10
Код буквы «О»: 11

Тогда слово «ОКНО» кодируется в виде последовательности «1101011»: коды всех букв записываются последовательно, один за другим.

Особую роль в теории кодирования играют *префиксные коды*. Код называется префиксным, если код никакой из букв не является началом строки (префиксом). Код, приведённый выше, является префиксным.

Префиксные коды являются «хорошими», потому что обеспечивают однозначность декодирования, то есть нельзя записать две различные последовательности букв, которые будут закодированы одинаковым способом.

Приведём пример кода, не являющегося префиксным.

Код буквы «К»: 0
Код буквы «Н»: 1
Код буквы «О»: 11

Если использовать такой код, то кодом слова «ОКНО» будет «110111», но точно так же будет кодироваться и слово «ОКОН». Более простой пример неоднозначного кодирования: слова «НН» и «О» кодируются одинаково: «11».

Кодовые слова могут иметь различную длину, при этом длина закодированного сообщения может оказаться различной. Например, рассмотрим следующий код.

Код буквы «К»: 00
Код буквы «Н»: 01
Код буквы «О»: 1

При использовании такого кода слово «ОКНО» будет закодировано строкой «100011», то есть будет короче, чем в ранее рассмотренном примере. Идея неравномерного кодирования (кодирования разных символов при помощи кодовых слов переменной длины) используется, например, при построении архиваторов.

Задание

Рассмотрим следующую строку:

КЛОУН УКОЛОЛ АКУЛУ ОКОЛО КОЛОКОЛА

В этой строке встречаются следующие буквы: А, К, Л, Н, О, У. Пробелы учитывать не будем, считаем, что строка не содержит пробелов.

Вам нужно придумать *двоичный префиксный* код для букв А, К, Л, Н, О, У, такой чтобы при использовании данного кода приведённая выше строка кодировалась бы как можно более короткой последовательностью символов 0 и 1.

Ответ нужно записать в форме шести строк, являющихся кодами букв А, К, Л, Н, О, У, именно в таком порядке. Приведённый код должен быть префиксным. При использовании данного кода строка «КЛОУН УКОЛОЛ АКУЛУ ОКОЛО КОЛОКОЛА» должна кодироваться в строку минимальной длины. Чем короче будет полученная при кодировании строка, тем больше баллов вы получите.

Например, если требуется закодировать строку «ОКНО», и в ответе нужно записать коды букв К, Н, О, то для получения закодированной строки минимальной длины ответ следует записать в таком виде.

00
01
1

Задание

Посчитаем, сколько раз встречается каждая буква в этой строке.

А — 2 раза.
К — 6 раз.
Л — 7 раз.
Н — 1 раз.
О — 9 раз.
У — 4 раза.

При неравномерном кодировании нужно самые часто встречающиеся буквы записывать короткими последовательностями, а редко встречающиеся буквы — длинными последовательностями.

Один из распространённых кодов — равномерный код, в котором все буквы кодируются словами равной длины. В данном случае можно выбрать длину кодового слова 3, можно составить 8 кодовых слов длины 3, из них в ответе нужно записать любые 6, например:

000
001
010
011
100
101

Такое решение будет оценено в 2 балла, т. к. не использует идею сокращения кодовых слов.

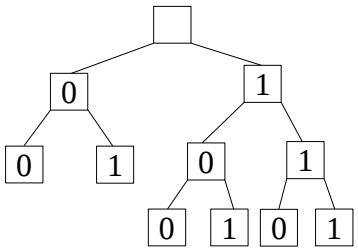
Можно уменьшить количество кодовых слов, сокращая их длину. Например, можно заменить два кодовых слова 000 и 001 на одно — 00, а кодовые слова 010 и 011 на слово 01. Мы получим набор из кодовых слов 00, 01, 100, 101, 110, 111. Для получения минимальной длины закодированного сообщения короткие слова нужно назначить часто встречающимся буквам: О и Л. Получим такой ответ:

100
101
00
110
01
111

Такой ответ не является самым лучшим, он оценивается в 8 баллов.

В этом ответе длины кодовых слов равны 2, 2, 3, 3, 3, 3, 3. Суммарная длина закодированного сообщения будет равна $2\times9 + 2\times7 + 3\times6 + 3\times4 + 3\times2 + 3\times1 = 71$ биту.

Для представления префиксных кодов используют двоичные деревья. В этом дереве в каждой вершине (кроме листа) есть два потомка. В вершинах (кроме корня) записаны цифры 0 и 1. Листья соответствуют кодовым словам, которые получаются записью всех цифр на пути от корня к листу. Построенному выше коду соответствует следующее дерево:

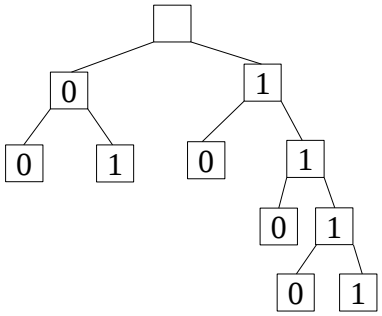


В этом дереве слева находятся кодовые слова 00 и 01 длины 2, а справа — кодовые слова 100, 101, 110, 111 длины 4.

Можно перебрать все деревья с 6 листами. Не будем рисовать все деревья, ограничимся указанием длин всех кодовых слов:

1, 2, 3, 4, 5, 5
1, 2, 4, 4, 4, 4
1, 3, 3, 3, 4, 4
2, 2, 4, 4, 4, 4
2, 2, 3, 3, 3, 3
2, 2, 2, 3, 4, 4

Если посчитать длину закодированного сообщения для каждого из этих кодов, то она будет наименьшей для последнего варианта: $2\times9 + 2\times7 + 2\times6 + 3\times4 + 4\times2 + 4\times1 = 68$ бит. Этому коду соответствует такое дерево:



Кодовые слова будут 00, 01, 10, 110, 1110, 1111. Ответ нужно записать в виде.

1110
00
01
1111
10
110

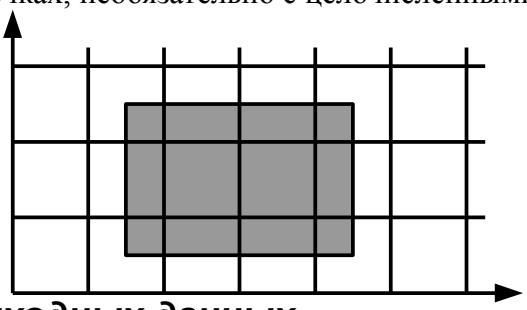
Задача 5. Плитка

Стена покрыта квадратной плиткой со стороной M см. На стену повесили картину, известны её ширина и высота. Определите максимальное количество плиток, которые могли оказаться частично или полностью закрыты картиной.

Первая строка входных данных содержит число M — сторону плитки. Вторая и третья строки содержат числа W и H — ширину и высоту картины. Все числа целые положительные, не превосходящие 2×10^9.

Программа должна вывести одно число — максимальное количество плиток, которые могли оказаться частично или полностью закрыты картиной.

Плитка считается закрытой картиной, если пересечение картины и плитки имеет ненулевую площадь, то есть касание картины и плитки не считается закрытием плитки. Стороны картины должны быть параллельны сторонам плитки. Углы картины могут находиться в произвольных точках, необязательно с целочисленными координатами.



Пример входных и выходных данных

Ввод	**Вывод**	**Примечание**
10 30 20	12	Пример соответствует картинке. Сторона плитки (сторона клетки на картинке) M = 10. Картина имеет ширину 30 см и высоту 20 см. Картина полностью или частично закрывает 12 плиток.

Система оценивания

Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 6 баллов.

Решение

Необходимо посчитать количество плиток, которое может закрывать картина по горизонтали и количество плиток, которое может закрывать картина по вертикали, затем перемножить найденные числа.

Если ширина картины равна *W*, сторона плитки равна *M*, то в случае, когда *W* делится на *M*, то есть $W = k\,M$, максимальное число закрытых плиток равно $k + 1$. Если же W не делится на M, то максимальное количество закрытых плиток равно $k + 2$, где k — целочисленное частное от деления *W* на *M*, т. к. можно закрыть отрезок из *k* подряд идущих плиток, и небольшой кусок плитки слева и справа от этого отрезка. Аналогично считается количество закрытых плиток по вертикали, и два числа перемножаются.

Пример решения на языке Python.

```
M = int(input())
W = int(input())
H = int(input())
num_w = W // M + 1
if W % M != 0:
    num_w += 1
num_h = H // M + 1
if H % M != 0:
    num_h += 1
print(num_w * num_h)
```

Задача 6. Лифт в бизнес-центре

Бизнес-центр представляет собой *N*-этажное здание, этажи пронумерованы от 1 до *N* снизу вверх. На каждом этаже работает ровно один сотрудник. Все сотрудники утром приезжают на парковку, которая расположена в подвальном помещении на один этаж ниже первого. Бизнес-центр оборудован лифтом, который вмещает неограниченное число людей, но вредный лифтёр сегодня готов отвезти всех сотрудников только на один какой-то этаж. С этого этажа сотрудники расходятся по своим этажам по лестнице, кто-то из сотрудников поднимается вверх, кто-то спускается вниз. На подъём на один этаж сотрудник тратит *A* секунд, спуск на один этаж занимает *B* секунд. Лифт тратит *C* секунд на подъём на один этаж.

Определите, на какой этаж необходимо отвезти сотрудников на лифте, чтобы все сотрудники попали на свои этажи как можно быстрее (с учётом того времени, которое они потратят на подъём на лифте).

Первая строка входных данных содержит число *N* — количество этажей в бизнес-центре. Следующие три строки содержат числа *A*, *B*, *C* — время, необходимое сотруднику на подъем на один этаж, на спуск на один этаж и время, необходимое лифту на подъём на один этаж. Все числа — целые положительные, не превосходящие 2×10^9, при этом $A > B$, $A > C$.

Программа должна вывести единственное целое число — номер этажа, на который должен ехать лифт, чтобы все сотрудники попали на свой этаж как можно быстрее.

Примеры входных и выходных данных

Ввод	**Вывод**	**Примечание**
6 20 10 5	4	В здании 6 этажей. Сотрудник поднимается на один этаж за 20 секунд, спускается за 10 секунд. Лифт поднимается на один этаж за 5 секунд. Чтобы всем сотрудникам быстрее добраться до своих мест, лифт едет на 4 этаж за 20 секунд. Сотрудник, который работает на 6 этаже,

		выходит из лифта и поднимается за 40 секунд, всего его путь занимает 60 секунд. Сотрудник, работающий на 1 этаже, спускается на 3 этажа, это занимает 20 + 30 = 50 секунд. При другом выборе этажа потребуется большее время.

Система оценивания

Решение, правильно работающее только для случаев, когда N не превосходит 100, будет оцениваться в 6 баллов.

Решение

Для решения на 6 баллов можно перебрать этаж, на который поднимается лифт. Пусть этот номер этажа равен i. Тогда для того, чтобы подняться на лифте на этаж номер i сотрудникам необходимо $i \times C$ секунд. Затем те сотрудники, которые спускаются вниз, потратят $B \times (i - 1)$ секунду, а те, кто поднимаются вверх — потратят $A \times (n - i)$ секунд. Итого все сотрудники разойдутся по своим этажам за время $i \times C + \max(B \times (i - 1), A \times (n - i))$. Переберём значение i и выберем тот этаж, для которого полученное время будет минимальным.

Пример такого решения.

```
n = int(input())
a = int(input())
b = int(input())
c = int(input())
ans = 0
best_time = n * a + 1
for i in range(1, n + 1):
    curr_time = i * c + max((i - 1) * b, (n - i) * a)
    if curr_time < best_time:
        best_time = curr_time
        ans = i
print(ans)
```

Это решение не уложится в ограничение по времени при больших n. Для решения на полный балл заметим, что в наилучшем решении сотрудники должны достигать первого и последнего этажа одновременно, то есть время, потраченное на подъём $A \times (n - i)$ равно времени, потраченному на спуск $B \times (i - 1)$. Приравняв эти времени и решив уравнение получим, что $i = (n \times A + B) / (A + B)$. Это было бы наилучшим ответом, если бы значение i могло быть дробным числом, но номер этажа может быть только целым числом. Поэтому нужно рассмотреть два значения i, округлив полученное действительное число вверх и вниз, для каждого из них посчитать ответ и выбрать наилучший.

Примере полного решения.

```
def time(floor):
    return floor * c + max((floor - 1) * b, (n - floor) * a)

n = int(input())
a = int(input())
b = int(input())
c = int(input())

floor = (n * a + b) // (a + b)
if time(floor) <= time(floor + 1):
    print(floor)
else:
    print(floor + 1)
```

Отметим, что в языках Pascal, C++, Java для получения полного балла необходимо проводить вычисления с использованием 64-битных целых чисел, т. к. при использовании обычных целых чисел происходит переполнение целочисленной переменной при вычислении.

Задача 7. Турнир

В турнире участвуют N команд. Турнир проводится по олимпийской системе (команды играют «на вылет», проигравшие команды выбывают из турнира, выигравшие проходят в следующий тур, ничьих не бывает). Число команд в этой задаче будет степенью двойки: $N = 2^k$.

Все команды пронумерованы числами от 1 до N. В первом туре играют команды с номерами 1 и 2, 3 и 4, 5 и 6 и т. д., всего играется $N / 2$ матчей. По результатам этих матчей команды выходят во второй тур. Во втором туре играют победители первой и второй игры первого тура, победители третьей и четвёртой игры первого тура и т. д. Они выходят в третий тур. В третьем круге играют вместе победители первой и второй игры второго тура, победители третьей и четвёртой игры второго тура и т. д.

Вам даны результаты всех матчей. Определите номер команды, которая стала победителем турнира.

В первой строке входных данных записано число N — количество команд, участвовавших в турнире. Оно является степенью двойки и может принимать значения от $2^0 = 1$ до $2^{16} = 65536$. Следующая N - 1 строка содержат результаты всех сыгранных матчей. Первые $N / 2$ строк из них являются результатами матчей первого тура, затем идёт $N / 4$ строк с результатами второго тура, $N / 8$ строк с результатами третьего тура и т. д.

Результат каждого матча является одним из двух возможных чисел: 1 или 2. Число 1 означает, что в матче выиграла первая команда (номер которой меньше), число 2 означает, что в матче выиграла вторая команда (номер которой больше).

Программа должна вывести одно число — номер победившей в турнире команды.

Пример входных и выходных данных

Ввод	**Вывод**
8 1 2 2 1 2 1 1	4

Система оценивания

Решение, правильно работающее только для случаев, когда $N \leq 8$, будет оцениваться в 4 балла.

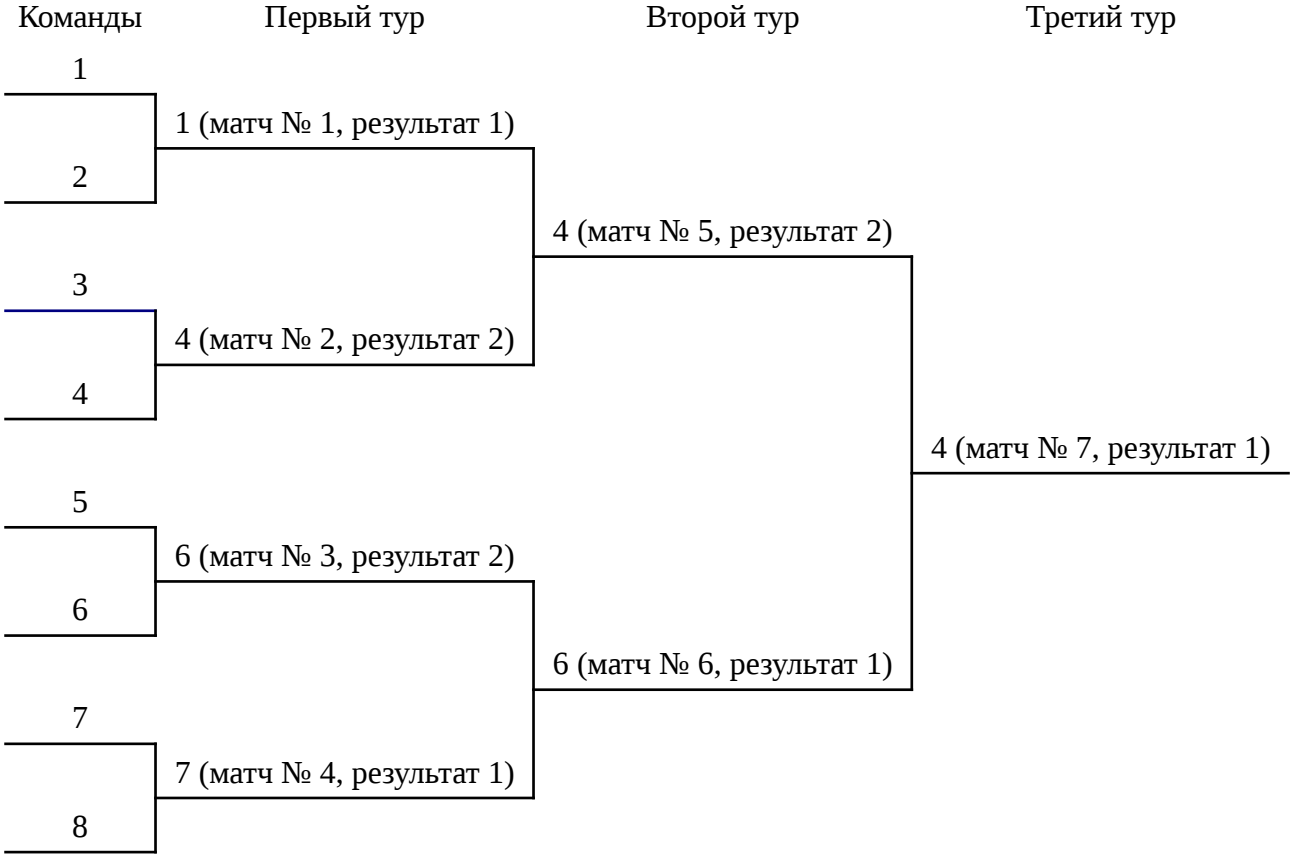
Пояснение к примеру

В следующей таблице нарисована схема турнира для примера из условия. В турнире участвовало 8 команд. Результаты матчей: 1, 2, 2, 1, 2, 1, 1.

В первом туре играли команды 1 и 2, 3 и 4, 5 и 6, 7 и 8. Результаты матчей первого тура: 1, 2, 2, 1, во второй тур вышли команды 1, 4, 6, 7.

Во втором туре играли команды 1 и 4, 6 и 7. Результаты матчей второго тура: 2, 1. В третий тур вышли команды 4 и 6.

В последнем третьем туре играют команды 4 и 6, результат матча: 1, поэтому победителем турнира является команда 4.



Решение

Эта задача является упражнением на использование массивов, в котором нужно смоделировать описанный процесс. Удобно использовать массивы переменной длины, например, в языке C++ удобно использовать контейнеры `vector`, а в языке Python — контейнер `list`.

Создадим массив из номеров команд, участвующих в турнире (teams), заполним его числами от 1 до N. Далее будем моделировать процесс турнира. В моделировании будет два цикла. Внешний цикл — по турам, этот цикл продолжается, пока в турнире не останется одна команда, то есть пока длина массива teams будет больше 1. Внутренний цикл — цикл по всем играм этого тура, это цикл с шагом 2. Длина этого цикла равна половине длины массива `teams`. Внутри этого цикла мы считываем результат игры. В игре участвуют две команды из массива teams: `teams[i]` и `teams[i + 1]`. Если результат игры равен 1, то в следующий тур выходит команда `teams[i]`, иначе — `teams[i + 1]`. Номер победившей команды добавляется в конец массива `left_teams` — это команды, вышедшие в следующий тур.

После окончания тура команды из массива left_teams переносятся в массив teams.

```
n = int(input())
teams = [i + 1 for i in range(n)]
while len(teams) > 1:
    left_teams = []
    for i in range(0, len(teams), 2):
        if int(input()) == 1:
            left_teams.append(teams[i])
        else:
            left_teams.append(teams[i + 1])
    teams = left_teams
print(teams[0])
```