Всероссийская олимпиада школьников по информатике, 2016/17 уч. год
Первый (школьный) этап, г. Москва
Разбор заданий для 9–11 классов

Задача 1. Покупка

Ручка стоила K рублей. Первого сентября стоимость ручки увеличилась ровно на P процентов. Определите, сколько ручек можно купить на S рублей после подорожания.

Программа получает на вход три целых положительных числа. Первое число K – стоимость ручки в рублях до подорожания. Второе число P – величина подорожания ручки в процентах. Третье число S – имеющаяся сумма денег. Числа K и S не превосходят 10^7, число P не превосходит 100.

Пример входных и выходных данных

Ввод	Вывод	Примечание
33 5 100	2	Ручка стоила 33 рубля. После подорожания на 5 % ручка будет стоить 34 рубля 65 копеек (заметим, что, поскольку первоначальная цена ручки была целым числом рублей, после подорожания стоимость ручки будет выражаться целым числом рублей и копеек). На 100 рублей после подорожания можно купить 2 ручки.

Система оценивания

Решение, правильно работающее только для случаев, когда числа K и S не превосходят 100, будет оцениваться в 60 баллов.

Решение

Посчитаем стоимость ручки в копейках после подорожания — она будет равна $K * 100 + K * P$ (один процент стоимости ручки составляет K копеек). Затем переведем S рублей в копейки (умножив на 100) и поделим нацело на стоимость ручки после подорожания.

Пример решения на языке Python

```
k = int(input())
p = int(input())
s = int(input())
k = 100 * k + p * k
print(100 * s // k)
```

Типичной ошибкой в этой задаче было использование действительных чисел для представления стоимости ручки после подорожания (если стоимость ручки хранить в рублях). Поскольку действительные числа представляются неточно, то после деления значения S на действительное число $K + K * P / 100$ может получиться не целое число, а число, которое чуть меньше правильного целого результата, и в результате преобразования частного к целому числу путём отбрасывания дробной части ответ получался на 1 меньше правильного. Такие решения набирали, как правило, 70 баллов.

Пример решения с использованием действительных чисел (70 баллов)

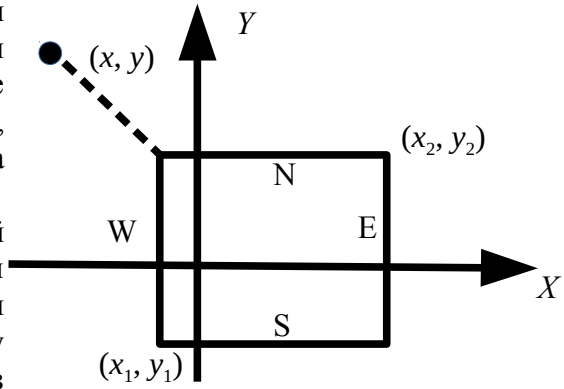
```
k = float(input())
p = int(input())
s = int(input())
k = k + k * p / 100
print(int(s / k))
```

Задача 2. Плот

Посередине озера плавает плот, имеющий форму прямоугольника. Стороны плота направлены вдоль параллелей и меридианов. Введём систему координат, в которой ось OX направлена на восток, а ось OY – на север. Пусть юго-западный угол плота имеет координаты (x_1, y_1), северо-восточный угол – координаты (x_2, y_2).

Пловец находится в точке с координатами (x, y). Определите, к какой стороне плота (северной, южной, западной или восточной) или к какому углу плота (северо-западному, северо-восточному, юго-западному, юго-восточному) пловцу нужно плыть, чтобы как можно скорее добраться до плота.

Программа получает на вход шесть чисел в следующем порядке: x_1, y_1 (координаты юго-западного угла плота), x_2, y_2 (координаты северо-восточного угла плота), x, y (координаты пловца). Все числа целые и по модулю не превосходят 100. Гарантируется, что $x_1 < x_2$, $y_1 < y_2$, $x \neq x_1$, $x \neq x_2$, $y \neq y_1$, $y \neq y_2$, координаты пловца находятся вне плота.



Если пловцу следует плыть к северной стороне плота, программа должна вывести символ «N», к южной – символ «S», к западной – символ «W», к восточной – символ «E». Если пловцу следует плыть к углу плота, нужно вывести одну из следующих строк: «NW», «NE», «SW», «SE».

Пример входных и выходных данных

Ввод	Вывод	Примечание
-1 -2 5 3 -4 6	NW	Картинка выше соответствует этому примеру.

Система оценивания

Решение, правильно работающее для случаев, когда ответом является одна из сторон плота «N», «S», «W», «E», будет оцениваться в 60 баллов.

Решение, правильно работающее для случаев, когда ответом является один из углов «NW», «NE», «SW», «SE», будет оцениваться в 40 баллов.

Решение

В этой задаче требовалось перебрать все возможные случаи расположения пловца относительно плота. Отметим на рисунке, что должна вывести программа для всех областей:

NW	N	NE
W	**плот**	E
SW	S	SE

Например, программа должна вывести NE при условиях $x > x_2$ и $y > y_2$, вывести N при условиях $y > y_2$ и $x_1 < x < x_2$ и т. д. Необходимо аккуратно разобрать все случаи.

Пример решения на языке Python

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
x = int(input())
y = int(input())
if x > x2 and y > y2:
   print("NE")
elif x > x2 and y < y1:
   print("SE")
elif x < x1 and y < y1:
   print("SW")
elif x < x1 and y > y2:
   print("NW")
elif y > y2:
   print("N")
elif y < y1:
   print("S")
elif x > x2:
   print("E")
else:
   print("W")
```

Но у задачи есть и более простое решение. Заметим, что программа должна вывести букву N во всех случаях, когда пловец находится севернее плота, в том числе и в областях NE и NW, то есть при условии $y > y_2$. Аналогично нужно вывести букву S всегда при выполнении условия $y < y_1$, букву W при условии $x < x_1$, букву E при условии $x > x_2$. При этом все «угловые» направления NW, NE, SW, SE получатся автоматически — сначала будет выведена одна из букв N или S, а затем одна из букв W или E. Такое решение содержит всего четыре условные инструкции if.

Пример решения на языке Python

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
x = int(input())
y = int(input())
ans = ""
if y > y2:
   ans += "N"
if y < y1:
   ans += "S"
if x < x1:
   ans += "W"
if x > x2:
   ans += "E"
print(ans)
```

Задача 3. Пакуем чемоданы!

Алёна собирает вещи в отпуск. С собой в самолёт она может взять ручную кладь и багаж. Для ручной клади у Алёны есть рюкзак, а для багажа – огромный чемодан.

По правилам перевозки масса ручной клади не должна превосходить S кг, а багаж может быть любой массы (за сверхнормативный багаж Алёна готова доплатить). Разумеется, наиболее ценные вещи – ноутбук, фотоаппарат, документы и т. д. – Алёна хочет положить в ручную кладь.

Алёна разложила все свои вещи в порядке уменьшения их ценности и начинает складывать наиболее ценные вещи в рюкзак. Она действует следующим образом – берёт самый ценный предмет, и если его масса не превосходит S, то кладёт его в рюкзак, иначе кладёт его в чемодан. Затем она берёт следующий по ценности предмет, если его можно положить в рюкзак, то есть если его масса вместе с массой уже положенных в рюкзак вещей не превосходит S, то кладёт его в рюкзак, иначе в чемодан, и таким же образом процесс продолжается для всех предметов в порядке убывания их ценности.

Определите вес рюкзака и чемодана после того, как Алёна сложит все вещи.

Первая строка входных данных содержит число S – максимально разрешённый вес рюкзака. Во второй строке входных данных записано число N – количество предметов. В следующих N строках даны массы предметов, сами предметы перечислены в порядке убывания ценности (сначала указана масса самого ценного предмета, затем масса второго по ценности предмета и т. д.). Все числа натуральные, число S не превосходит 2×10^9, сумма весов всех предметов также не превосходит 2×10^9. Значение N не превосходит 10^5.

Программа должна вывести два числа – вес рюкзака и вес чемодана (вес пустого рюкзака и чемодана не учитывается).

Пример входных и выходных данных

Ввод	Вывод	Примечание
20 5 6 10 5 2 3	18 8	Максимально возможная масса рюкзака 20 кг. Дано 5 предметов весом 6, 10, 5, 2, 3. Сначала предмет весом 6 кладётся в рюкзак, затем предмет весом 10 тоже кладётся в рюкзак. Предмет весом 5 нельзя положить в рюкзак, так как тогда вес рюкзака станет 21 кг, поэтому предмет весом 5 кладётся в чемодан. Затем предмет весом 2 кладётся в рюкзак, а предмет весом 3 – в чемодан. Вес рюкзака 6 + 10 + 2 = 18, вес чемодана 5 + 3 = 8.

Система оценивания

Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов.

Решение

Эта задача оказалась самой простой для большинства участников. Действительно, в этой задаче нет ни «подводных камней», как в первой задаче, ни объёмного разбора случаев, как во второй задаче. Необходимо просто реализовать тот алгоритм, который описан в условии: завести две переменные для хранения массы рюкзака и массы чемодана, считывать очередное число и прибавлять его либо к массе рюкзака, либо к массе чемодана, следя за тем, чтобы масса чемодана не превышала значения S.

Пример решения на языке Python

```
s = int(input())
n = int(input())
s1 = 0
s2 = 0
for i in range(n):
   a = int(input())
   if s1 + a <= s:
       s1 += a
   else:
       s2 += a
print(s1, s2)
```

Типичная ошибка в этой задаче — использование строгого неравенства s1 + a < s вместо нестрогого s1 + a <= s. Такие решения набирали 60 баллов.

Задача 4. Туристический налог

Для пополнения бюджета в стране Авалон, известной своими горными туристическими маршрутами, ввели новый налог для туристов. Величина налога пропорциональна длине маршрута, но, поскольку маршрут проходит по горам и пройденное расстояние, зависящее от высоты спуска и подъёма, подсчитать сложно, налог считается без учёта высоты, то есть величина налога пропорциональна горизонтальному перемещению, совершённому туристической группой. Кроме того, в силу старинного обычая все туристические группы должны перемещаться по горам Авалона строго с запада на восток.

Турфирма хочет сэкономить на налоге, поэтому она хочет разработать туристический маршрут с минимальной величиной налога. При этом, поскольку маршрут является горным, он должен содержать подъём в гору и спуск с горы, то есть на маршруте должна быть точка, которая находится строго выше начала и конца маршрута.

Турфирма составила карту гор Авалона, содержащую информацию о высоте гор при передвижении с запада на восток. Высоты гор измерены в точках через равные расстояния. Найдите на данной карте гор Авалона туристический маршрут минимальной длины, удовлетворяющий условию наличия подъёма и спуска.

Первая строка входных данных содержит число N – количество точек на карте гор Авалона. Следующие N строк содержат информацию о высоте гор в данных N точках при движении с запада на восток. Все числа натуральные, не превосходящие 10^5.

Программа должна вывести два числа – номер точки начала маршрута и номер точки окончания маршрута. Точки нумеруются от 1 до N. Если маршрута, удовлетворяющего условиям, не существует, программа должна вывести одно число 0.

Пример входных и выходных данных

Ввод	Вывод	Примечание
7 18 10 15 20 20 10 3	3 6	Дано 7 точек с высотами 18, 10, 15, 20, 20, 10, 3. Самый короткий маршрут, содержащий подъём и спуск, – это 15, 20, 20, 10. Он начинается в точке номер 3 и заканчивается в точке номер 6.
3	0	Высота гор монотонно убывает, поэтому искомого

9 8 5		маршрута не существует.

Система оценивания

Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов.

Решение

Эта задача, напротив, оказалась наиболее сложной задачей олимпиады, на полный балл её решило около 3 % участников.

Для начала можно попробовать перебрать все возможные ответы, то есть все возможные начальные и конечные точки маршрутов, при помощи двух вложенных циклов. Затем найдём максимальное значение высоты на этом участке, для чего понадобится ещё один цикл (в приведённом ниже решении вместо третьего цикла используется функция max для среза списка, то есть для рассматриваемого фрагмента). Если значение максимума на маршруте строго больше значений крайних элементов, то этот маршрут удовлетворяет условию задачи. Осталось только выбрать самый короткий маршрут из таких (то есть маршрут, для которого минимальная разница для индексов конца и начала маршрута). Такое решение имеет алгоритмическую сложность $O(N^3)$, то есть время работы такой программы растёт пропорционально N^3 и набирает 40 баллов.

Пример решения на языке Python (40 баллов)

```
n = int(input())
a = [int(input()) for i in range(n)]
ans1 = 0
ans2 = 10 ** 9
for i in range(0, n):
   for j in range(i + 2, n):
       m = max(a[i:j + 1])
       if a[i] < m > a[j] and j - i <= ans2 - ans1:
           ans2 = j
           ans1 = i
if ans2 != 10 ** 9:
   print(ans1 + 1, ans2 + 1)
else:
   print(0)
```

Сложность этого решения можно уменьшить до $O(N^2)$, если максимум на отрезке считать быстрее. Для этого можно заметить, что при увеличении правой границы j на 1 к рассматриваемому отрезку добавляется один элемент, поэтому можно хранить текущее значение максимума на отрезке и при увеличении j новый добавляемый к отрезку элемент сравнивать с максимумом, обновляя максимум при необходимости. Но такое решение тоже не получает максимальный балл.

Для того, чтобы придумать правильное и эффективное решение заметим, что ответ (высоты наилучшего маршрута) всегда имеет вид, как в примере (15, 20, 20, 10) — все значения кроме двух крайних равны, а два крайних меньше их. Иначе можно сократить маршрут, оставив только два крайних значения, меньших наибольшего значения на маршруте. Поэтому можно изучать только последовательности равных идущих подряд элементов массива, среди всех таких последовательностей нужно выбрать

последовательность наименьшей длины, причём перед и после этой последовательности должны быть меньшие значения. При этом алгоритм должен иметь сложность O(*N*), иначе программа также не сможет уложиться в ограничение по времени в 1 секунду.

Решение можно реализовать и без использования массива для хранения всех данных чисел. Будем считывать числа по одному, при этом будем использовать следущие переменные:

`last_height` — значение последнего считанного элемента.

`last_len` — количество последних считанных элементов, равных `last_height`.

`prev_height` — значение элемента, который был перед последовательностью последних элементов равных `last_height`.

Пусть значение нового считанного элемента равно `h`. Если `h` = `last_height`, то новый элемент продолжает последовательность элементов равных `last_height`, поэтому увеличим значение `last_len` на 1. Если же новый элемент неравен `last_height`, то этот элемент начинает новуй последовательность равных элементов, поэтому запишем его значение в переменную `last_height`, переменной `last_len` присвоим значение 1, а старое значение `last_height` переместится в переменную `prev_height`. Но сначала проверим, является ли последовательность элементов равных `last_height`, удовлетворяющей условию задачи, что означает выполнение неравенств `prev_height` < `last_height` > `h`. Если эти неравенства верны, а также если длина последовательности `last_len` наименьшая (для этого будем также хранить наименьшую известную длину последовательности, удовлетворяющую условию задачи в переменной `best_len`), то запомним ответ и обновим значение переменной `best_len`.

Пример решения на языке Python (100 баллов)

```
n = int(input())
ans1 = 0
ans2 = 0
last_height = 10 ** 9
prev_height = 10 ** 9
last_len = 0
best_len = 10 ** 9
for i in range(1, n + 1):
   h = int(input())
   if h == last_height:
       last_len += 1
   else:
       if prev_height < last_height > h and last_len < best_len:
           best_len = last_len
           ans1 = i - last_len - 1
           ans2 = i
       prev_height = last_height
       last_height = h
       last_len = 1
if ans1 == 0:
   print(0)
else:
   print(ans1, ans2)
```

Задача 5. Делимость

Сегодня в школе на уроке математики проходят делимость. Чтобы продемонстрировать свойства делимости, учитель выписал на доске все целые числа от 1 до N в несколько групп, при этом если одно число делится на другое, то они обязательно оказались в разных группах. Например, если взять $N = 10$, то получится 4 группы.

Первая группа: 1.

Вторая группа: 2, 7, 9.

Третья группа: 3, 4, 10.

Четвёртая группа: 5, 6, 8.

Вы уже догадались, что, поскольку любое число делится на 1, одна группа всегда будет состоять только из числа 1, но в остальном подобное разбиение можно выполнить различными способами. От вас требуется определить минимальное число групп, на которое можно разбить все числа от 1 до N в соответствии с приведённым выше условием.

Программа получает на вход одно натуральное число N, не превосходящее 10^9, и должна вывести одно число – искомое минимальное количество групп.

Пример входных и выходных данных

Ввод	**Вывод**
10	4

Система оценивания

Решение, правильно работающее только для случаев, когда N не превосходит 20, будет оцениваться в 20 баллов.

Решение, правильно работающее только для случаев, когда N не превосходит 10^3, будет оцениваться в 40 баллов.

Решение, правильно работающее только для случаев, когда N не превосходит 10^4, будет оцениваться в 60 баллов.

Решение

Решение на 20 баллов можно получить, если разобрать случаи маленьких значений N и для каждого из них посчитать ответ.

Пример решения на языке Python (20 баллов)

```
n = int(input())
if n == 1:
   print(1)
elif n <= 3:
   print(2)
elif n <= 7:
   print(3)
elif n <= 15:
   print(4)
else:
   print(5)
```

Для полного решения достаточно заметить, что ответ увеличивается на 1, когда N равно степени двойки: 2, 4, 8, 16 и т. д. Действительно, пусть максимальная степень двойки, не превосходящая числа N, равна 2^k. Тогда ответ будет как минимум равен $k + 1$, поскольку среди рассматриваемых N чисел есть $k + 1$ степень двойки, включая $1 = 2^0$, все эти степени

двойки делятся друг на друга, поэтому они должны оказаться в разных группах, то есть ответ будет не меньше, чем $k + 1$. Покажем, что $k + 1$ группы достаточно, то есть все числа от 1 до N можно разбить на $k + 1$ группу. Для этого пронумеруем группы от 0 до k и в группу с номером i отнесём числа от 2^i до 2^{i+1}-1 (в последнюю группу попадут число от 2^k до N). Тогда в одной группе не может оказаться двух чисел, одно из которых нацело делится на другое, т. к. частное от деления наибольшего числа в группе на наименьшее число меньше 2.

Пример решения на языке Python (100 баллов)

```
n = int(input())
ans = 1
p = 1
while 2 * p <= n:
   p *= 2
   ans += 1
print(ans)
```

Частичные баллы (40 или 60) по этой задаче можно было получить, если промоделировать процесс распределения чисел по группам. Будем рассматривать числа по одному в порядке возрастания, для каждого числа n будем перебирать его делители. Это можно сделать за O(n), если перебирать все делители циклом от 1 до n - 1 (или до n / 2), а можно за O($n^{½}$), если перебирать делители только до $n^{½}$, и для каждого делителя d числа n рассматривать также парный ему делитель n / d. Число n будем помещать в группу с минимальным номером, которая не содержит делителей числа n, а если таких групп нет, то добавляем число n в новую группу. Такое решение помимо неэффективности ещё и достаточно тяжело в реализации.

Пример решения на языке C++ (60 баллов)

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

using namespace std;

vector<vector<int> > nums;
vector<int> used;

void check(int d)
{
   for (int i = 0; i < nums.size(); ++i)
       if (!used[i] && find(nums[i].begin(),
                            nums[i].end(), d) != nums[i].end())
       {
           used[i] = 1;
           return;
       }
}

int main()
{
   int n;
   cin >> n;
   nums.resize(1);
```

```
    nums[0].push_back(1);
    used.resize(1);
    for (int i = 2; i <= n; ++i)
    {
        fill(used.begin(), used.end(), 0);
        for (int d = 1; d * d <= n; ++d)
            if (i % d == 0)
            {
                check(d);
                check(i / d);
            }
        if (find(used.begin(), used.end(), 0) == used.end())
        {
            nums.push_back(vector<int>(1, i));
            used.push_back(0);
        }
    }
    cout << nums.size() << endl;
}
```