Задача A. Клетки с общей стороной

Самый маленький номер будет иметь клетка под заданной, следующий по величине - клетка слева от нее, затем - справа, и, наконец, клетка сверху от заданной.

Сначала рассмотрим, какие номера клеток вообще могут иметь с X общую границу.

Номера клеток слева и справа отличаются от числа X на 1 в меньшую и большую сторону соответственно, а номера клеток снизу и сверху - на M. Номера клеток, которые потенциально могли бы быть соседями, мы вычислили.

Для клеток снизу и сверху от заданной, очевидно, достаточно проверить существование. Это значит, что нужно убедиться, что полученные значения больше 0 и не больше $N*M$ соответственно.

Но, если в таблице 3*4 взять клетку 5, то она стоит в левом столбце, а значит слева от нее ничего нет. Аналогично, клетка 4 не может иметь соседнюю клетку справа. Для проверки этих ситуаций легко понять, что все клетки на правой границе поля имеют номера, кратные M, а клетки на левой границе - номера, на единицу большие, чем числа, кратные M.

Решение на языке программирования Python может выглядеть так:

```
n = int(input())
m = int(input())
x = int(input())

if x - m > 0:
    print(x - m)
if (x - 1) % m != 0:
    print(x - 1)
if x % m != 0:
    print(x + 1)
if x+m <= n*m:
    print(x + m)
```

Задача B. Сладкий санаторий

В этой задаче требуется выяснить, какое количество сладостей каждого вида не удастся поровну разделить между K мальчиками, то есть, каков остаток от деления количества сладостей каждого вида на K.

Решение сводится к циклу, в котором N раз считывается число и находится остаток от его деления на K.

Можно решить эту задачу с использованием массива для хранения ответов. Сначала будет происходить считывание исходных данных и запись ответов в массив, а затем в новом цикле содержимое массива будет выведено в поток вывода (на экран). Такое решение правильно работает на 100% тестов.

Однако, поскольку потоки ввода и вывода работают независимо друг от друга, то более оптимальным считается решение, в котором массив не используется: после подсчета каждого нового остатка его можно сразу отправлять в поток вывода. Тогда все решение состоит из одного цикла, в котором считывается число и сразу выводится его остаток от деления на K. При локальном выполнении такой программы на экране компьютера потоки вывода и ввода будут отображаться по очереди. Но для тестирующей системы ответ программы будет находиться в потоке вывода, а сам по себе этот поток будет полностью идентичен тому, который получился бы при использовании массива. Поэтому такое решение тоже работает на 100% тестов.

Пример решения на языке программирования Python:

```
k = int(input())
n = int(input())

for i in range(n):
    print(int(input()) % k, end = " ")
```

Задача C. Тарифный план для Интернета

Это несколько усложненная задача на поиск минимума. Программа должна посчитать стоимость заданного трафика при всевозможных тарифных планах, и выбрать тот план, у которого наименьшая стоимость, либо наименьшая абонентская плата при равной стоимости, либо меньший порядковый номер при равных стоимости и абонентской плате.

Для удобства введем переменные:

- стоимость трафика при рассмотрении очередного тарифного плана - st,
- объем потребляемого трафика - M,
- размер абонентской платы abR рублей и abK копеек в месяц,
- включенный в абонентскую плату трафик - T,
- стоимость каждого дополнительного мегабайта - $megR$ и $megK$ в рублях и копейках соответственно.

Теперь рассчитаем стоимость трафика M, выполняя одновременно с этим перевод всех рублей в копейки:

$st = abR*100 + abK + (megR*100 + megK)*(M - T)$, если $M > T$, или

$st = abR*100 + abK$ если $M <= T$.

Мы научились определять стоимость данного трафика для одного тарифного плана, значит можем после считывания данных о тарифном плане сразу узнать эту стоимость. Иными словами, хранить данные о тарифных планах в каких-либо массивах не требуется.

Теперь посмотрим, как найти ответ на задачу - номер минимального тарифного плана и его стоимость в рублях и копейках.

До начала цикла, в котором будет производится считывание тарифных планов и расчет стоимости трафика для них, заведем переменные:

- минимальная стоимость - $minst$,
- номер тарифа, для которого рассчитана минимальная стоимость - $minK$,
- размер абонентской платы в копейках для тарифа с минимальной стоимостью - $minab$.

Начальные значения для этих переменных можно задать двумя способами: либо прочитать характеристики первого тарифного плана, вычислить соответствующие значения и сохранить их в указанных переменных, либо присвоить им сверхбольшие числа, которые не могли бы получиться в тарифных планах, определенных условиями данной задачи. Поскольку в условии сказано, что для хранения всех целочисленных значений достаточно стандартных 32-битных типов данных, можно считать, что таким сверхбольшим числом для данной задачи будет число $2 * 10^9$.

В приведенных решениях показаны оба способа - первый в решении на языке программирования Pascal, второй - в решении на языке программирования Python.

Каждый раз в цикле, рассчитав стоимость трафика для очередного тарифного плана, будем сравнивать ее с хранящейся в $minst$ найденной ранее минимальной стоимостью. Если стоимость при текущем тарифном плане окажется меньше, то будем обновлять значение всех трех переменных с приставкой min, поскольку мы нашли новый минимальный тарифный план.

В случае, если стоимость для очередного тарифного плана совпадет с ранее найденной минимальной стоимостью, нужно будет выполнить сравнение текущей и минимальной абонентских плат, а если и они равны, перейдем к сравнению номеров тарифных планов.

После окончания цикла в переменных $minK$ и $minst$ будет храниться ответ на задачу. Но по условию стоимость должна быть показана двумя числами - в рублях и копейках, поэтому при выводе ответа нужно выполнить соответствующее разделение.

В решении на языке программирования Pascal для удобства записи программы использована функция chet, которая рассчитывает стоимость трафика по выбранному тарифу на основе формул, приведенных выше.

Код авторского решения на языке программирования Pascal:

```
function chet(M, k, abR, abK, T, megR, megK: integer): integer;
var
  st: integer;
begin
  st := abR * 100 + abK;
  if M > T then
    st := st + (megR * 100 + megK) * (m - t);
  chet := st;
end;

var
  i,n, M, k, abR, abK, T, megR, megK, st, minst, minab, minK: integer;

begin
  readln(M);
  readln(N);
  readln(k, abR, abK, T, megR, megK);
  minst := chet(M, k, abR, abK, T, megR, megK);
  minab := abR * 100 + abK;
  minK := k;
  for i := 2 to n do
  begin
    readln(k, abR, abK, T, megR, megK);
    st := chet(M, k, abR, abK, T, megR, megK);
    if (minst > st) or (minst = st) and (abR * 100 + abK < minab) or
      (minst = st) and (abR * 100 + abK = minab) and (k < minK) then
    begin
      minst := st;
      minab := abR * 100 + abK;
      minK := k;
    end;
  end;
  writeln(mink);
  writeln(minst div 100, ' ', minst mod 100);
end.
```

Для удобства начинающих программистов, изучающих язык программирования Python, приводим запись решения задачи на этом языке программирования с несколько иными названиями переменных:

```
m = int(input())
n = int(input())

mn = 2000000000
minab = 2000000000
mink = 2000000000
for i in range(n):
    k, ab1, ab2, t, ext1, ext2 = (int(x) for x in input().split())
    cost = ab1*100 + ab2 + (max(0, m - t)*(ext1*100 + ext2))
    if cost < mn:
        mn = cost
        minab = ab1*100 + ab2
        mink = k
    if cost == mn and ab1*100 + ab2 < minab:
        minab = ab1*100 + ab2
```

```
        mink = k
    if cost == mn and ab1*100 + ab2 == minab and k < mink:
        mink = k

print(mink)
print(mn // 100, mn % 100, sep = ' ')
```