Всероссийская олимпиада школьников по информатике, 2015/16 уч. год
Первый (школьный) этап, г. Москва
Решения заданий для 9–11 классов

Задача 1. Считалка

Условие

Для выбора водящего в детской игре N человек становятся в круг, после чего произносится считалка. На первом слове считалки указывается на первого человека в кругу, на втором слове – на второго человека и т. д. После N-го человека снова идёт первый человек (все люди в кругу пронумерованы числами от 1 до N, круг зацикливается, после человека с номером N идёт человек с номером 1).

Всего в считалке M слов. Определите, на какого человека придётся последнее слово считалки.

Программа получает на вход два целых положительных числа. Первое число N – количество людей в кругу. Второе число M – количество слов в считалке. Оба числа не превосходят 10^9.

Программа должна вывести одно целое число от 1 до N – номер человека в кругу на которого придётся последнее слово считалки.

Пример входных и выходных данных

Ввод	**Вывод**
10 25	5

Система оценивания

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

Решение

Ответом является остаток от деления числа M на число N, за единственным исключением – если остаток равен нулю, то есть M делится на N, то считалка остановится на последнем человеке и программа должна вывести значение N, а не 0. Это нужно рассмотреть при помощи одного условия if.

Пример решения задачи на языке Python:

```
N = int(input())
M = int(input())
if M % N == 0:
    print(N)
else:
    print(M % N)
```

Типичная ошибка в этой задаче — не рассмотрен случай, когда M делится на N, такое решения набирало 60 баллов. Решение, которое использовало цикл длиной M, также набирало около 60 баллов, поскольку не укладывалось в ограничение по времени работы программы.

Задача 2. Конфеты

Условие

На столе стоят три вазы с конфетами. В левой вазе лежат A конфет, в средней вазе лежат B конфет, в правой вазе лежат C конфет. Лена съедает одну конфету из левой вазы, затем – одну конфету из средней вазы, затем из правой, средней, левой, средней, правой, средней и т. д. (слева направо, затем налево, опять направо и т.д.)

Если Лена хочет взять конфету из какой-то вазы, а конфет там нет, она расстраивается и идёт спать. Определите, сколько конфет съест Лена.

Программа получает на вход три целых неотрицательных числа A, B, C – количество конфет в левой, средней, правой вазе. Сумма трёх данных чисел не превосходит 2×10^9.

Пример входных и выходных данных

Ввод	**Вывод**	**Примечание**
3 3 3	7	Лена съест конфеты из левой, средней, правой, средней, левой, средней, правой вазы. После этого она захочет съесть конфету из средней вазы, но в ней уже не осталось конфет.

Система оценивания

Решение, правильно работающее только для случаев, когда входные числа не превосходят 10, будет оцениваться в 40 баллов.

Решение, правильно работающее только для случаев, когда входные числа не превосходят 10 000, будет оцениваться в 70 баллов.

Решение

Решение, полностью моделирующее процесс взятия конфет, то есть уменьшающее число конфет на 1 в каждой вазе в соответствии с описанным в условии алгоритмом, не будет укладываться в ограничение по времени работы программы и должно набирать 70 баллов.

Для получения полного решения заметим, что процесс взятия конфет содержит цикл «левая ваза, средняя ваза, правая ваза, средняя ваза», который затем повторяется. За один проход такого цикла число A уменьшается на 1, число B уменьшается на 2, число C уменьшается на 1. Посчитаем, сколько раз будет выполнен цикл — это минимум из чисел A, $[B / 2]$ и C (под записью $[B / 2]$ подразумевается целая часть от деления B на 2, то есть операция целочисленного деления). Запишем количество проходов цикла в переменную k и уменьшим значение переменных A и C на k, а значение переменной B на $2k$. За k исполнений цикла суммарно будет взято $4k$ конфет.

Следующий проход цикла не будет выполнен полностью. Посмотрим на значение переменных A, B, C в том порядке, в котором берутся конфеты из соответствующих ваз. Если $A = 0$, то нельзя на следующем шаге взять конфету из первой вазы, и ответом будет $4k$. Если $B = 0$, то будет взята конфета из первой вазы, но во второй вазе конфеты кончились, поэтому ответ будет $4k + 1$. Если же $C = 0$, то, аналогично, можно взять еще две конфеты из левой и средней вазы, и ответ будет $4k + 2$. Наконец, если все эти условия не выполнены, то ответ будет $4k + 3$.

Пример решения задачи на языке Python (операция «//» в Python обозначает целочисленное деление):

```
a = int(input())
b = int(input())
c = int(input())
k = min(a, b // 2, c)
```

```
a -= k
b -= 2 * k
c -= k
if a == 0:
    print(4 * k)
elif b == 0:
    print(4 * k + 1)
elif c == 0:
    print(4 * k + 2)
else:
    print(4 * k + 3)
```

Задача 3. Расписание кружка

Условие

Володе очень понравились задачи олимпиады по информатике, поэтому он решил ходить на занятия кружка по программированию. Придя на первое занятие кружка, он узнал, что занятия будут проходить еженедельно в один и тот же день недели. Помогите Володе составить календарь занятий до конца года – определите даты всех занятий, начиная с первого занятия и до конца года.

Программа получает на вход два числа, записанных в разных строках: номер месяца и номер дня месяца, когда проходит первое занятие. Номер месяца может быть одним из четырёх возможных чисел – 9, 10, 11, 12. Номер дня месяца – число от 1 до 30 для сентября и ноября (месяцы с номерами 9 и 11) или от 1 до 31 для октября и декабря (месяцы с номерами 10 и 12).

Программа должна вывести даты всех занятий кружка до конца года в хронологическом порядке, по одной дате в строке, сначала месяц, затем день месяца, через пробел. Занятия проходят еженедельно, в тот же день недели, что и первое занятие. Формат вывода дат такой же, как в условии. Считайте, что каникулы отсутствуют, а последнее занятие может происходить в любой день декабря, в том числе и 31 числа.

Пример входных и выходных данных

Ввод	**Вывод**
11 20	11 20 11 27 12 4 12 11 12 18 12 25

Решение

В этой задаче нет больших чисел и требуется всего лишь организовать правильный цикл по дням. Будем хранить в двух переменных m и d номер месяца и номер дня месяца. Затем в цикле будем увеличивать значение d на 7, что соответствует переходу на следующую неделю. Далее нужно аккуратно обработать переходы в следующий месяц. В сентябре 30 дней, поэтому если $m = 9$, а $d > 30$, то произошел переход в октябрь, поэтому нужно присвоить значение $m := 10$, а значение d уменьшить на 30. Аналогично, если $m = 10$, а $d > 31$, то произошел переход в ноябрь, поэтому нужно присвоить значение $m := 11$, а значение d уменьшить на 31. Если $m = 11$, а $d > 30$, то произошел переход в декабрь, нужно присвоить значение $m := 12$, а значение d уменьшить на 31. Наконец, если $m = 12$, а $d > 31$, то

произошел переход в новый год и нужно завершить работу программы.

Пример решения задачи на языке Python:

```
m = int(input())
d = int(input())
while d <= 31:
    print(m, d)
    d += 7
    if m == 9 and d > 30:
        m = 10
        d -= 30
    if m == 10 and d > 31:
        m = 11
        d -= 31
    if m == 11 and d > 30:
        m = 12
        d -= 30
```

Задача 4. Кратное трём число

Дано число. В этом числе необходимо изменить одну цифру таким образом, чтобы новое число делилось на 3 и было бы максимально возможным. В исходном числе нужно обязательно изменить одну цифру, даже если исходное число уже делилось на 3.

Программа получает на вход одно длинное натуральное число. Длина числа может достигать 100 цифр.

Программа должна вывести другое натуральное число, удовлетворяющее условиям:

1. Новое число должно отличаться от данного ровно одной цифрой.
2. Новое число должно делиться на 3.
3. Новое число должно быть максимально возможным из всех таких чисел.

Пример входных и выходных данных

Ввод	**Вывод**
123	723

Система оценивания

Решение, правильно работающее только для случаев, когда входное число содержит не более трёх цифр, будет оцениваться в 30 баллов.

Решение, правильно работающее только для случаев, когда входное число содержит не более девяти цифр, будет оцениваться в 60 баллов.

Решение

Несмотря на то, что речь в этой задаче идет о числе, эта задача является задачей на умение обрабатывать строки, поскольку многие языки программирования не имеют поддержки длинных целых чисел. Поэтому входную последовательность чисел нужно считать в строку и работать с ней, как со строкой.

Прежде всего определим остаток от деления исходного числа на 3. Для этого нужно посчитать сумму цифр исходного числа (то есть сумму отдельных символов строки), и взять остаток от деления суммы на 3. Если остаток от деления на 3 суммы цифр равен 0, то нужно увеличить какую-то цифру числа на 3, 6 или 9, так как в этом случае число уже делится на 3 и нужно изменить одну цифру так, чтобы остаток от деление на 3 не изменился. Если остаток равен 1, то одну цифру числа нужно увеличить на 2, 5 или 8. Если же остаток равен 3, то одну цифру числа нужно увеличить на 1, 4 или 7.

Поскольку нам нужно сделать новое число максимально большим, то нужно увеличить как можно более левую цифру исходного числа. Будем перебивать все цифры исходного числа слева направо, пока не найдем первую цифру, которую можно увеличить на нужную величину, при этом цифру нужно увеличить как можно больше (после увеличения эта цифра станет равна 7, 8 или 9).

Наконец, возможна ситуация, когда ни одной цифры увеличить нельзя, такое возможно, например, когда число состоит из одних цифр 9, или для числа 888, или для числа 7878. В этом случае нужно уменьшить последнюю цифру числа так, чтобы число стало кратным 3. Например, число 7878 будет преобразовано в число 7875.

Пример решения задачи на языке Python:

```
N = input()
sum = 0
for digit in N:
    sum += int(digit)
inc = 3 - sum % 3
for i in range(len(N)):
    if int(N[i]) + inc <= 9:
        d = int(N[i]) + inc
        while d + 3 <= 9:
            d += 3
        N = N[:i] + str(d) + N[i + 1:]
        print(N)
        break
else:
    d = int(N[-1])
    if inc == 1:
        d -= 2
    elif inc == 2:
        d -= 1
    else:
        d -= 3
    N = N[:-1] + str(d)
    print(N)
```

Это решение является наиболее эффективным, его вычислительная сложность составляет O(n), где n — длина данного числа. Но поскольку длина числа не превосходит 100 цифр, можно придумать и менее эффективное решение, которое все равно наберет максимальный балл. Будем перебирать все цифры исходного числа по одной и будем менять каждую цифру числа на все возможные цифры от 0 до 9. В результате мы получим новое число, которое должно быть не равно исходному числу и должно делиться на 3. Делимость на 3 можно проверить, посчитав остаток от деления суммы цифр числа, а в языке Python есть длинная целочисленная арифметика, поэтому можно преобразовать строку к типу int и взять остаток от деления на 3. В результате мы будем получать какие-то новые числа, которые будут храниться, как строки, отличающиеся от исходной строки одним символом и кратные 3. Из этих строк нужно взять максимальную.

Пример подобного решения на языке Python:

```
n = input()
ans = 0
digits = '0123456789'
for i in range(len(n)):
    for digit in digits:
        new = int(n[:i] + digit + n[i + 1:])
        if digit != n[i] and new % 3 == 0:
            ans = max(ans, new)
print(ans)
```

Задача 5. Минимальное произведение

Условие

Дана последовательность из N целых чисел (они могут быть положительными, отрицательными или равными 0). Необходимо выбрать из этих чисел два числа так, чтобы их произведение было как можно меньшим (не рассматриваются квадраты данных чисел, но можно выбрать произведение двух различных элементов последовательности, равных друг другу).

В первой строке входных данных записано целое число N, $2 \le N \le 10^5$ – количество данных чисел. Следующие N строк содержат сами числа, не превосходящие по модулю 40 000.

Программа должна вывести единственное целое число – наименьшее возможное произведение двух различных элементов этой последовательности.

Пример входных и выходных данных

Ввод	**Вывод**
3 1 -3 2	-6

Система оценивания

Тесты к этой задаче разбиты на четыре группы, приведённые в таблице. При этом решение обязательно должно проходить тест из условия задачи.

Кол-во баллов	Ограничение на N	Ограничение на значение членов последовательности
20	$2 \le N \le 100$	Неотрицательные
20	$2 \le N \le 100$	Положительные, отрицательные или ноль
20	$100 < N \le 10^5$	Неотрицательные
40	$100 < N \le 10^5$	Положительные, отрицательные или ноль

Решение

Если в числе есть как положительные, так и отрицательные числа, то можно получить отрицательное произведение, и его нужно сделать как можно большим по модулю. Для этого нужно перемножить самое большое положительное число на самое большое по модулю отрицательное число, то есть на наименьшее число. Если все данные числа положительные, то нужно перемножить два наименьших числа, а если все числа отрицательные — то два наибольших числа, то есть два наименьших по модулю числа.

Таким образом, для решения задачи можно найти наибольшее число (запишем его в переменную max1), второе по величине число (то есть то, которое будет наибольшим, если вычеркнуть одно наибольшее число, запишем его в переменную max2) и два наименьших числа (min1 и min2). Дальше нужно рассмотреть три произведения — min1 × max1, min1 × min2 и max1 × max2 и выбрать из них наименьшее. Пример решения на языке Python:

```
n = int(input())
min1 = 40001
min2 = 40001
max1 = -40001
max2 = -40001
for i in range(n):
    x = int(input())
    if x > max1:
        max2 = max1
        max1 = x
    elif x > max2:
        max2 = x
    if x < min1:
        min2 = min1
        min1 = x
    elif x < min2:
        min2 = x
print(min(min1 * max1, min1 * min2, max1 * max2))
```

Такое решение имеет сложность O(n), то есть работает за время, пропорциональное числу входных данных, и даже не сохраняет все считанные данные в памяти.

Можно вместо поиска двух минимумов и двух максимумов считать все данные числа, сохранить их в массиве и отсортировать массив. При этом нужно использовать какой-либо из алгоритмов быстрой сортировки, имеющий сложность O($n \log n$), или же использовать встроенную сортировку языка программирования. После этого два наименьших элемента будут в первых двух элементах массива, а два наибольших элемента — в двух последних. На языке Python можно обратиться к двум первым элементам массива, как A[0] и A[1], а к двум последним — как к A[-2] и A[-1]. Пример подобного решения:

```
n = int(input())
A = [int(input()) for i in range(n)]
A.sort()
print(min(A[0] * A[-1], A[0] * A[1], A[-2] * A[-1]))
```

Решение, содержащее два вложенных цикла, то есть перебирающее все пары элементов, имеет сложность O(n^2) и получало 40 баллов. Пример такого неэффективного решения:

```
n = int(input())
A = [int(input()) for i in range(n)]
ans = 2000000000
for i in range(len(A)):
    for j in range(i + 1, len(A)):
        ans = min(ans, A[i] * A[j])
print(ans)
```