# Луппов Иван Витальевич

## Балл: 108

### Задача №1

#### Условие

Вася наблюдает, как строители копают фундамент для нового дома. Землю они вывозят одинаковыми самосвалами. По каким-то своим причинам всю выбранную землю вывозят со стройки в тот же день, в который ее выбрали, то есть даже за одним кубометром все равно вечером приедет самосвал. Ещё одна странность, которую заметил Вася, что первый в день самосвал всегда наполняли максимум на половину.

Васе известно, сколько дней копали фундамент, какой объем земли вывозили каждый день и сколько кубометров помещается в самосвал. Вася любил считать всё, что можно, и в этот раз он решил считать самосвалы, но отвлекался на домашнюю работу и теперь боится, что посчитал не верно. Напишите программу, которая поможет Васе проверить общее количество самосвалов, которые вывозили грунт.

В первой строке на вход программы подаются два натуральных числа: количество дней $t$, в течение которых копали фундамент, $1 < t \leq 10^3$ и объем земли $v$, который вмещается в один самосвал, $1 < v \leq 10^2$. На следующей строке на вход программы поступает $n$ целых неотрицательных чисел: $a_1$, $a_2$, …, $a_n$. $a_i$ – сколько земли выбрали в $i$-й день, $1 < a_i \leq 10^5$

Выведите одно целое число – количество требуемых самосвалов.

### Пример

| Входные данные | Выходные данные |
|---|---|
| 2 10<br>10 15 | 4 |

*Комментарий к примеру*. Строители работали два дня, в первый день они извлекли 10 кубометров грунта, и увезли двумя самосвалами по 5 кубов каждый, на следующий день тоже понадобилось 2 самосвала: первый вывез 5, второй 10 кубов. Итого потребовалось 4 самосвала.

#### Исходный код

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

int main(){
    long long n, v, ans = 0, a;
    cin >> n >> v;
    double f = (double)v / 2;
    for(int i = 0; i < n; ++i){
        cin >> a;
        if(!i){
            ans += (double)a / f;
            ceil(ans);
            if(a - (double)(ans) * f != 0) ans++;
        }
        else ans += a / v + !(!(a % v));
    }
    cout << ans;
}
```

### Задача №2

#### Условие

В кассе, продающей билеты по цене 500 р., нет возможности расплатиться картой, а в начале работы у кассира для сдачи есть только $m$ купюр по 500 р. Перед открытием у кассы собрались посетители, у каждого человека в очереди есть только одна купюра. У части – 500 р., у части – 1000 р. Сколькими

способами можно выстроить посетителей в очередь так, чтобы к моменту обслуживания посетителя с купюрой в 1000 рублей у кассира всегда была сдача?

Напишите программу, которая решит эту задачу.

На вход программы подаются три целых неотрицательных числа:

***n*** - количество человек с купюрой 500 р,

***k*** - количество человек с купюрой 1000 р

***m*** - количество купюр по 500 р у кассира к началу работы кассы.

Каждое число меньше 10.

Выведите одно целое число – сколькими способами можно выстроить очередь.

Пример

| Входные данные | Выходные данные |
|---|---|
| 1 2 1 | 4 |

*Комментарий к примеру*. Пусть есть посетители **A**, **B**, и **C**. У **A** и **B** по 1000 рублёвой купюре, а у **C** – купюра в 500 рублей. Тогда их можно выстроить так:

**ABC** – нельзя, т.к. единственная купюра 500р. уйдет на сдачу посетителю **A**, а для посетителя **B** сдачи не будет.

**ACB** – можно, т.к. единственная купюра 500р. уйдет на сдачу посетителю **A,** затем посетитель **C** заплатит купюрой 500р., и для покупателя **B** будет сдача.

**BAC** – нельзя, т.к. единственная купюра 500р уйдет на сдачу посетителю **B**, а для посетителя **A** сдачи не будет.

**BCA** – можно

**CAB** – можно

**CBA** – можно

Есть только 4 возможных варианта.

**Исходный код**

```
#include <iostream>
using namespace std;

long long con = 0;

void soch(long long a, long long b, long long c){
    if(!a && !b){
        con++;
        return;
    }
    if(a) soch(a - 1, b, c + 1);
    if(b && c) soch(a, b - 1, c - 1);
    else return;
}

int main(){
    long long a, b, c;
    cin >> a >> b >> c;
    soch(a, b, c);
    long long ans = con;
    while(a){
        ans *= a;
        a--;
    }
    while(b){
        ans *= b;
        b--;
    }
    cout << ans;
}
```

## Задача №3

**Условие**

Исполнитель получает на вход натуральное число X (не превышающее $10^6$). По этому числу, точнее по его представлению в восьмеричной системе счисления, строится новое число Y по следующим правилам.

В восьмеричном представлении числа X предпоследняя цифра увеличивается на 1 (гарантируется, что в восьмеричном представлении X числа больше 2-х цифр). Например, $695_{10} = 12\mathbf{6}7_8$ -> $12\mathbf{7}7_8 = 703_{10}$.

Если предпоследняя цифра 7, тогда предпоследняя цифра становится 0, а последняя изменяется по следующему принципу: четная увеличивается н 1, а нечетная уменьшается на 1. Например, последняя цифра нечетная $697_{10} = 12\mathbf{7}\boldsymbol{1}_8$ -> $12\mathbf{0}\boldsymbol{0}_8 = 640_{10}$, последняя цифра четная $698_{10} = 12\mathbf{7}\boldsymbol{2}_8$ -> $12\mathbf{0}\boldsymbol{3}_8 = 643_{10}$.

Введем понятие расстояния

Oh = | Исходное_число – Полученное_число |

Напишите программу, которая будет считать наибольшее расстояние Oh для чисел из заданного интервала [A,B] и наибольшее исходное число, для которого оно было вычислено.

На вход программы подаётся два целых числа A и B ($10 \le A \le B \le 1\,000\,000$), записанных через пробел.

Программа должна вывести два числа наибольшее расстояние Oh и через пробел исходное число, для которого оно было посчитано.

| Входные данные | Вывод | Примечание |
|---|---|---|
| 694 698 | 57 697 | $\lvert 694 - 702 \rvert = 8$ ($12\mathbf{6}6_8 - 12\mathbf{7}6_8$)<br>$\lvert 695 - 703 \rvert = 8$ ($12\mathbf{6}7_8 - 12\mathbf{7}7_8$)<br>$\lvert 696 - 541 \rvert = 55$ ($12\mathbf{7}\mathbf{0}_8 - 12\mathbf{0}\mathbf{1}_8$)<br>$\lvert 697 - 640 \rvert = 57$ ($12\mathbf{7}\mathbf{1}_8 - 12\mathbf{0}\mathbf{0}_8$)<br>$\lvert 698 - 643 \rvert = 55$ ($12\mathbf{7}\mathbf{2}_8 - 12\mathbf{0}\mathbf{3}_8$) |
| 693 695 | 8 695 | $\lvert 693 - 701 \rvert = 8$ ($12\mathbf{6}5_8 - 12\mathbf{7}5_8$)<br>$\lvert 694 - 702 \rvert = 8$ ($12\mathbf{6}6_8 - 12\mathbf{7}6_8$)<br>$\lvert 695 - 703 \rvert = 8$ ($12\mathbf{6}7_8 - 12\mathbf{7}7_8$) |

**Исходный код**

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;


int main(){
    long long a, b, mx = 0, z = 0;
    cin >> a >> b;
    for(int i = a; i <= b; ++i){
        long long now = i, nw = 0, c = 1;
        while(now){
            nw += (now % 8) * c;
            now /= 8;
            c *= 10;
        }
        nw += 10;
        if(nw % 100 >= 80){
            nw -= 80;
            if(nw % 2 == 0) nw++;
            else nw--;
        }
        long long ans = 0;
        c = 1;
        while(nw){
            ans += (nw % 10) * c;
```

```
            nw /= 10;
            c *= 8;
        }
        if(abs(i - ans) >= mx){
            mx = abs(i - ans);
            z = i;
        }
    }
    cout << mx << " " << z;
}
```

## Задача №4

### Условие

Сообщение, которое передают по каналу связи, состоит из чисел, записанных в десятичной системе счисления. Каждое число состоит из шести знаков, а его двоичная запись оканчивается тремя нулями. При передаче сообщение было засорено посторонними шумами: числами, отличающимися от тех, что были в сообщении. Найдите изначальное количество чисел в сообщении.

*Формат ввода*

В строке вводится сначала целое число ***n*** – количество чисел в сообщении ( $n \leq 1000$), затем ***n*** натуральных чисел, все числа отделены друг от друга одним или несколькими пробелами.

*Формат вывода*

Вывести одно целое число – количество достоверных сигналов в сообщении.

## Пример

| Входные данные | Выходные данные |
|---|---|
| 3 100128 4356 234064 | 2 |

### Исходный код

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

long long c(long long n){
    if(!(n / 10)) return 1;
    return(1 + c(n / 10));
}


int main(){
    long long n, a, ans = 0;
    cin >> n;
    for(int i = 0; i < n; ++i){
        cin >> a;
        if(c(a) != 6) continue;
        if(!(a % 2) && !(a / 2 % 2) && !(a / 4 % 2)) ans++;
    }
    cout << ans;
}
```

## Задача №5

### Условие

Несколько агентов пересылают кодовые сообщения в Центр. Сообщение каждого агента представляет собой несколько слов, к каждому из которых приписан его идентификатор, состоящий из одной буквы. Сообщения записаны по очереди. Найдите сообщение, содержащее больше всего слов.

*Формат ввода*

В первой строке вводится сначала целое число ***n*** – количество слов ($n \leq 1000$), затем в ***n*** следующих строках записано по слову. Слова состоят только из строчных латинских букв.

*Формат вывода*

Вывести одно целое число – длину сообщения (количество слов в сообщении), содержащего больше всего слов.

**Примеры**

| Входные данные | Выходные данные |
|---|---|
| 4<br>bcd<br>bcfd<br>bcdfe<br>abc | 2 |
| 3<br>abd<br>abdc<br>bvd | 1 |

**Исходный код**

```
#include <iostream>
#include <vector>
#include <cmath>
#include <string>
using namespace std;


int main(){
    long long n, ans = 0, c = 1;
    cin >> n;
    vector<string> s(n);
    if(n) cin >> s[0];
    for(int i = 1; i < n; ++i){
        cin >> s[i];
        if(s[i][s[i].size() - 1] == s[i - 1][s[i - 1].size() - 1]) c++;
        else{
            ans = max(c, ans);
            c = 1;
        }
    }
    cout << ans;
}
```

## Задача №6

**Условие**

Робот может выполнять команды «Поиск дефектов», «Движение», «Подготовка» и «Ремонт». Из-за конструктивных особенностей на робота наложены некоторые ограничения. Два раза подряд можно выполнить только команду «Движение». Команда «Ремонт» может быть выполнена только на следующем шаге после команды «Подготовка».

Напишите программу, которая определит, сколько существует выполнимых последовательностей команд длиной n, если до начала выполнения программы робот выполнил команду «Подготовка».

На вход программе подается натуральное число n ($n \le 15$) – количество команд.

Вывести целое число – количество выполнимых последовательностей команд длиной n.

**Пример**

| Ввод | Вывод |
|---|---|

| | |
|---|---|
| 2 | 8 |

**Исходный код**

```
#include <iostream>
#include <vector>
#include <cmath>
#include <string>
using namespace std;

long long ans = 0;


void p(long long n, long long h, vector<char> &s){
    if(h == n){
        ans++;
        return;
    }
    if(s[s.size() - 1] != 'p'){
        s.push_back('p');
        p(n, h + 1, s);
        s.pop_back();
    }
    if(s[s.size() - 1] != 'd' || (s.size() > 1 && s[s.size() - 2] != 'd')){
        s.push_back('d');
        p(n, h + 1, s);
        s.pop_back();
    }
    if(s[s.size() - 1] != 'g'){
        s.push_back('g');
        p(n, h + 1, s);
        s.pop_back();
    }
    if(s[s.size() - 1] == 'p'){
        s.push_back('r');
        p(n, h + 1, s);
        s.pop_back();
    }
}


int main(){
    long long n;
    cin >> n;
    vector<char> a;
    a.push_back('p');
    p(n, 0, a);
    cout << ans;
}
```

## Задача №7

**Условие**

Беспилотник летит над большим озером в спокойную погоду. В него заложена программа, в соответствии с которой он изменяет высоту полета один раз в минуту. В программе могут быть следующие команды:

«Выше» – беспилотник поднимается на 0,5 м.

«Ниже» – беспилотник опускается на 0,5 м.

«Сохранение высоты» - беспилотник сохраняет высоту до следующей команды. Таким образом, все перемещения беспилотника происходят с шагом 0.5 м. Набор высоты или снижение происходят за время, пренебрежимо малое по сравнению с минутой. Если беспилотник касается воды, он выходит из строя. Зная начальную высоту и продолжительность полета, найдите, сколькими способами можно задать последовательность команд, не приводящую к разрушению беспилотника, после выполнения которой он окажется в заданном диапазоне высот.

На вход программе подаются вещественное число $h_0$ ($1 \le h_0 \le 100$) – начальная высота квадрокоптера, целое число t (t<=20) – продолжительность полета, два вещественных числа l и h ($1 \le l \le h \le 100$)– нижняя и верхняя граница требуемого диапазона высот. Все вещественные числа даны с точностью до 0,5 м.

Программа должна вывести одно целое число – количество способов.

## Пример

| Ввод | Вывод |
|---|---|
| 2 2 2.5 3 | 3 |

### Исходный код

```
#include <iostream>
#include <vector>
#include <cmath>
#include <string>
using namespace std;

long long ans = 0;
double mn, mx;


void p(long long n, long long h, double h0){
    if(h == n){
        if(h0 >= mn && h0 <= mx) ans++;
        return;
    }
    if(0.5 * (n - h - 1) >= mn - h0)p(n, h + 1, h0);
    if(0.5 * (n - h - 1) >= h0 + 0.5 - mx)p(n, h + 1, h0 + 0.5);
    if(h0 > 0.5 && 0.5 * (n - h - 1) >= mn - h0 + 0.5) p(n, h + 1, h0 - 0.5);
}


int main(){
    long long t;
    double h0;
    cin >> h0 >> t >> mn >> mx;
    p(t, 0, h0);
    cout << ans;
}
```