

# Колышев Глеб Викторович

Балл: 140

## 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:  $a_1, a_2, \dots, a_n$ .  $a_i$  – 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 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 "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double
#define pb push_back
#define sz(v) (int)v.size()

#define int ll

#define LOCAL

#ifdef LOCAL
    const int maxn = 20;
    #define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
    #define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
```

```

if (k > n) return 0;
int ans = 1;
int t = 1;
int tmp = n;
while (tmp > n - k) {
    ans *= tmp;
    ans /= t;
    --tmp;
    ++t;
}
return ans;
}

int fact (int x) {
int ans = 1;
for (int i = 1; i <= x; ++i) ans *= i;
return ans;
}

int binpow (int a, int n) {
if (n == 0) return 1;
if (n % 2) return a * binpow (a, n - 1);
int b = binpow (a, n / 2);
return b * b;
}

int bit (int msk, int i) {
return (msk >> i) & 1;
}

signed main() {
ios_base::sync_with_stdio (false);
cin.tie (nullptr);
cout.tie (nullptr);
int n, h;
cin >> n >> h;
vector<int> goo (n);
cinv (goo);
int ans = 0;
for (auto& x: goo) {
    int t = h / 2;
    ++ans;
    x -= t;
    if (x <= 0) continue;
    ans += (x + h - 1) / h;
}
cout << ans;

//system("pause");

}

```

## 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 "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double
#define pb push_back
#define sz(v) (int)v.size()

#define int ll

#define LOCAL

#ifdef LOCAL
    const int maxn = 20;
#define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
#define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
    if (k > n) return 0;
    int ans = 1;
    int t = 1;
    int tmp = n;
    while (tmp > n - k) {
        ans *= tmp;
        ans /= t;
        --tmp;
        ++t;
    }
    return ans;
}

int fact (int x) {
    int ans = 1;
    for (int i = 1; i <= x; ++i) ans *= i;
    return ans;
}

int binpow (int a, int n) {
    if (n == 0) return 1;
    if (n % 2) return a * binpow (a, n - 1);
    int b = binpow (a, n / 2);
    return b * b;
}
```

```

int bit (int msk, int i) {
    return (msk>>i) & 1;
}

int check (int msk, int m, int n, int k) {
    int bal = m;
    int mn = m;
    int cnt1 = 0, cnt2 = 0;
    for (int i = 0; i < n + k; ++i) {
        if (bit (msk, i)) ++cnt1;
        else ++cnt2;
    }
    if (!(cnt1 == n && cnt2 == k)) return 0;
    for (int i = 0; i < n + k; ++i) {
        if (bit (msk, i)) ++bal;
        else --bal;
        mn = min (mn, bal);
    }
    mn = min (mn, bal);
    return mn >= 0;
}

signed main() {
    ios_base::sync_with_stdio (false);
    cin.tie (nullptr);
    cout.tie (nullptr);
    int n, k, m;
    cin >> n >> k >> m;
    int M = (int)pow (2, n + k);
    int ans = 0;
    for (int msk = 0; msk < M; ++msk) {
        ans += check (msk, m, n, k);
    }
    ans *= fact (n) * fact (k);
    cout << ans;
    //system("pause");
}

```

### 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,  $695_{10} = 1267_8 \rightarrow 1277_8 = 703_{10}$ .

Если предпоследняя цифра 7, тогда предпоследняя цифра становится 0, а последняя изменяется по следующему принципу: четная увеличивается на 1, а нечетная уменьшается на 1. Например, последняя цифра нечетная  $697_{10} = 1271_8 \rightarrow 1200_8 = 640_{10}$ , последняя цифра четная  $698_{10} = 1272_8 \rightarrow 1203_8 = 643_{10}$ .

Введем понятие расстояния

$$Oh = | \text{Исходное\_число} - \text{Полученное\_число} |$$

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  $B$  ( $10 \leq A \leq B \leq 1\,000\,000$ ), 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	$ 694 - 702  = 8$ ( $1266_8 - 1276_8$ ) $ 695 - 703  = 8$ ( $1267_8 - 1277_8$ ) $ 696 - 541  = 55$ ( $1270_8 - 1201_8$ ) $ 697 - 640  = 57$ ( $1271_8 - 1200_8$ ) $ 698 - 643  = 55$ ( $1272_8 - 1203_8$ )
693 695	8 695	$ 693 - 701  = 8$ ( $1265_8 - 1275_8$ ) $ 694 - 702  = 8$ ( $1266_8 - 1276_8$ ) $ 695 - 703  = 8$ ( $1267_8 - 1277_8$ )

### Исходный код

```

#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double
#define pb push_back
#define sz(v) (int)v.size()

// #define int ll

#define LOCAL

#ifndef LOCAL
    const int maxn = 20;
#define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
#define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
    if (k > n) return 0;
    int ans = 1;
    int t = 1;
    int tmp = n;
    while (tmp > n - k) {
        ans *= tmp;
        ans /= t;
        --tmp;
        ++t;
    }
    return ans;
}

int fact (int x) {
    int ans = 1;
    for (int i = 1; i <= x; ++i) ans *= i;
    return ans;
}

int binpow (int a, int n) {
    if (n == 0) return 1;
    if (n % 2) return a * binpow (a, n - 1);
    int b = binpow (a, n / 2);
    return b * b;
}

int kek (int x) {
    int t = x;
    vector<int> ans;
    while (x) {
        ans.pb (x % 8);
    }
}

```

```

    x /= 8;
}
if (ans[1] == 7) {
    ans[1] = 0;
    if (ans[0] % 2) {
        --ans[0];
    } else {
        ++ans[0];
    }
} else --ans[1];
int tmp = 0;
for (int i = 0; i < sz (ans); ++i) {
    tmp += ans[i] * binpow (8, i);
}
return abs (t - tmp);
}

signed main() {
    ios_base::sync_with_stdio (false);
    cin.tie (nullptr);
    cout.tie (nullptr);
    int l, r;
    cin >> l >> r;
    int ans = 0, res = 1;
    for (int i = l; i <= r; ++i) {
        int k = kek (i);
        if (k >= ans) ans = k, res = i;
    }
    cout << ans << ' ' << res;

    //system("pause");

}

```

## 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 "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double

```

```

#define pb push_back
#define sz(v) (int)v.size()

#define int ll

#define LOCAL

#ifdef LOCAL
    const int maxn = 20;
#define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
#define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
    if (k > n) return 0;
    int ans = 1;
    int t = 1;
    int tmp = n;
    while (tmp > n - k) {
        ans *= tmp;
        ans /= t;
        --tmp;
        ++t;
    }
    return ans;
}

int fact (int x) {
    int ans = 1;
    for (int i = 1; i <= x; ++i) ans *= i;
    return ans;
}

int kek (int x) {
    int t = x;
    int res = 0;
    while (t) {
        ++res;
        t /= 10;
    }
    vector<int> ans;
    while (x) {
        ans.pb (x % 2);
        x /= 2;
    }
    int cnt = 0;
    for (int i = 0; i < min (311, sz (ans)); ++i) cnt += ans[i] == 0;
    return cnt == 3 && res == 6;
}

signed main() {
    ios_base::sync_with_stdio (false);
    cin.tie (nullptr);
    cout.tie (nullptr);
    int n;
    cin >> n;
    vector<int> goo (n);
    cinv (goo);
    int ans = 0;
    for (auto& x: goo) ans += kek (x);
    cout << ans;

    //system("pause");

}

```

## 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 bcd bcfd bcdfe abc	2
3 abd abdc bvd	1

### Исходный код

```
//#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double
#define pb push_back
#define sz(v) (int)v.size()

#define int ll

#define LOCAL

#ifdef LOCAL
    const int maxn = 20;
    #define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
    #define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
    if (k > n) return 0;
    int ans = 1;
    int t = 1;
    int tmp = n;
    while (tmp > n - k) {
        ans *= tmp;
        ans /= t;
        --tmp;
        ++t;
    }
    return ans;
}
```

```
int fact (int x) {
    int ans = 1;
    for (int i = 1; i <= x; ++i) ans *= i;
    return ans;
}
```

```
signed main() {
    ios_base::sync_with_stdio (false);
    cin.tie (nullptr);
    cout.tie (nullptr);
    int n;
    cin >> n;
    vector<string> goo (n);
    cinv (goo);
    int cnt = 0, ans = 0;
    char prev = '#';
    for (auto& x: goo) {
        if (x.back() == prev) ++cnt;
        else {
            ans = max (ans, cnt);
            cnt = 1;
            prev = x.back();
        }
    }
    ans = max (ans, cnt);
    cout << ans;
    //system("pause");
}
```

## Задача №6

### Условие

Робот может выполнять команды «Поиск дефектов», «Движение», «Подготовка» и «Ремонт». Из-за конструктивных особенностей на работа 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  $n$  ( $n \leq 15$ ) – количество команд.

Вывести целое число – количество выполнимых последовательностей команд длиной  $n$ .

### Пример

Ввод	Вывод
2	8

### Исходный код

```
//#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double
#define pb push_back
#define sz(v) (int)v.size()

#define int ll

#define LOCAL
```

```

#ifdef LOCAL
    const int maxn = 20;
#define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
#define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
    if (k > n) return 0;
    int ans = 1;
    int t = 1;
    int tmp = n;
    while (tmp > n - k) {
        ans *= tmp;
        ans /= t;
        --tmp;
        ++t;
    }
    return ans;
}

int fact (int x) {
    int ans = 1;
    for (int i = 1; i <= x; ++i) ans *= i;
    return ans;
}

signed main() {
    ios_base::sync_with_stdio (false);
    cin.tie (nullptr);
    cout.tie (nullptr);
    int n;
    cin >> n;
    vector <vector<int>> dp (n + 1, vector<int> (4, 0)); // 0 - search, 1 - move, 2 - premode, 3 - recovery
    dp[0][2] = 1;
    dp[1][0] = 1;
    dp[1][1] = 1;
    dp[1][3] = 1;
    for (int i = 2; i <= n; ++i) {
        for (int j = 0; j < 4; ++j) {
            for (int k = 0; k < 4; ++k) {
                if (k == j && k != 1) continue;
                if (j == 3 && k != 2) continue;
                dp[i][j] += dp[i - 1][k];
            }
        }
    }
    int ans = 0;
    for (auto& v: dp[n]) {
        ans += v;
    }
    cout << ans;

    //system("pause");
}

```

## 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  $h_0$  ( $1 \leq h_0 \leq 100$ ) – начальная высота квадрокоптера, целое число  $t$  ( $t \leq 20$ ) – продолжительность полета, два вещественных числа  $l$  и  $h$  ( $1 \leq l \leq h \leq 100$ ) – 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 "stdafx.h"
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <string>

#define ll long long
#define cinv(v) for (auto& x: v) cin >> x
#define fr first
#define se second
#define all(v) v.begin(), v.end()
#define ld long double
#define pb push_back
#define sz(v) (int)v.size()

#define int ll

#define LOCAL

#ifdef LOCAL
    const int maxn = 20;
#define dbg(x) cerr << #x << " : " << x << '\n';
#else
    const int maxn = 2e5 + 10;
#define dbg(x)
#endif

using namespace std;

int C (int n, int k) {
    if (k > n) return 0;
    int ans = 1;
    int t = 1;
    int tmp = n;
    while (tmp > n - k) {
        ans *= tmp;
        ans /= t;
        --tmp;
        ++t;
    }
    return ans;
}

int fact (int x) {
    int ans = 1;
    for (int i = 1; i <= x; ++i) ans *= i;
    return ans;
}

signed main() {
    ios_base::sync_with_stdio (false);
    cin.tie (nullptr);
    cout.tie (nullptr);
    ld h, t, l, r;
    cin >> h >> t >> l >> r;
    vector<vector<int>>> dp (t + 1, vector<int> (500));
    h *= 2, l *= 2, r *= 2;
    dp[0][h] = 1;
    for (int i = 1; i <= t; ++i) {
        for (int j = 1; j < 499; ++j) {
```

```
    dp[i][j] += dp[i - 1][j - 1] + dp[i - 1][j] + dp[i - 1][j + 1];
}
}
int ans = 0;
for (int i = 1; i <= r; ++i) {
    ans += dp[t][i];
}
cout << ans;

//system("pause");

}
```