Проскурников Юрий Юрьевич

Балл: 118

Задача №1

Условие

На складе хранится K болтов M12x60 и L шпилек M12x40. Для изготовления одного изделия требуется P крепёжных элементов, из них не менее трёх болтов.

Определите, можно ли использовать весь имеющийся крепёж при изготовлении изделий данного вида.
Входные данные: 3 целых положительных числа - K, L, P, каждое меньше 1000, записанные через пробел.

Выходные данные: возможное количество изготовленных изделий либо 0, если использовать весь крепёж не получится.

Исходные данные	Результат
10 110 40	3
15 150 20	0

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

int main() {
    int k, l, p;
    cin >> k >> l >> p;
    if (p > k + l || k < 3)
        cout << 0;
    else {
        if ((k + l) % p)
            cout << 0;
        else {
            if ((k + l) / p > k / 3)
                cout << 0;
            else
                cout << ((k + l) / p);
        }
    }
    return 0;
}
```

Задача №2

Условие

Один из производителей электроники, фирма "Хардстронг", разработала прототип нового вычислительного устройства. Оно позволяет обрабатывать 16-разрядные целые положительные числа и способно выполнять над ними 3 операции: побитовый сдвиг влево на k разрядов (k <= 15), вычитание 1024 и вычитание 1.

В случае побитового сдвига старшие разряды числа, выходящие за пределы разрядной сетки, теряются. Младшие разряды заполняются нулями. Например, сдвиг числа 3 на 2 разряда даст новое значение 12, а сдвиг числа 8193 на 7 даст 128.

В случае получения в результате какой-либо операции отрицательного числа либо нуля выполнение программы завершится исключением (ошибкой).

Выполнение каждой из трёх операций занимает определённое время, не зависящее от значений операндов: сдвиг - 4 такта, вычитание 1024 - 2 такта, вычитание 1 - один такт.

Для заданных целых положительных чисел A и B, каждое меньше 65536, требуется определить минимальное количество тактов для получения числа B из числа A. В случае, если с помощью описанного вычислителя число B из числа A получить невозможно, вывести -1.

Исходные данные	Результат
4 8	4
2049 1024	3

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

const int MOD = 131072;

map<int, bool> used;
map<int, int> cnt;

int main() {
    int n, m, tmp, k = 0;
    cin >> n >> m;
    if (n == m) {
        cout << 0;
        return 0;
    }
    queue<int> q;
    q.push(n);
    for (int k = 0; k < 50000; k++) {
        tmp = q.front();
        q.pop();
       // cout << tmp << endl;
        int x = tmp;

        if (tmp - 1024 > 0) {
            cnt[tmp - 1024] = (cnt[tmp - 1024] ? min(cnt[tmp - 1024], cnt[tmp] + 2) : cnt[tmp] + 2);
            used[tmp - 1024] = 1;
            q.push(tmp - 1024);
        }

        if (tmp - 1 > 0) {
            cnt[tmp - 1] = (cnt[tmp - 1] ? min(cnt[tmp - 1], cnt[tmp] + 1) : cnt[tmp] + 1);
            used[tmp - 1] = 1;
            q.push(tmp - 1);
```

```
        }
        for (int i = 0; i < 16; i++) {
            x = (x * 2) % MOD;
            cnt[x] = (cnt[x] ? min(cnt[x], cnt[tmp] + 4) : cnt[tmp] + 4);
            used[x] = 1;
           /* if (x == m) {
                cout << cnt[tmp] << ' ' << cnt[m] << endl;
            }*/
            q.push(x);
        }

    }
    if (!used[m])
        cout << -1;
    else
        cout << cnt[m];
    return 0;
}
```

Задача №3

Условие

Дано множество отрезков на плоскости. Требуется определить, сколько параллелограммов можно построить из этих отрезков. При этом отрезки можно перемещать, но нельзя менять их размер и угол наклона к осям. Один отрезок можно включать только в один параллелограмм.

Входные данные: в первой строке натуральное N < 100 - количество отрезков. Далее в каждой строке по 4 целых числа, записанных через пробел: x1, y1, x2, y2 - координаты начала и конца отрезка.

Выходные данные: натуральное число - количество параллелограммов.

Исходные данные	Результат
5 0 0 5 0 0 1 5 1 0 0 1 1 1 1 2 2 1 2 5 2	1
4 0 0 0 2 0 0 1 2 0 0 2 2 0 0 2 0	0

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

struct vect {
    ll x, y;
};

bool used[100];

bool kol(vect a, vect b) {
    return (a.x * b.y - b.x * a.y == 0) && ((a.x * a.x + a.y * a.y) == (b.x * b.x + b.y * b.y));
}

int main() {
    int n, ans = 0;
    cin >> n;
    vect v[n];
    for (int i = 0; i < n; i++) {
        ll x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        v[i].x = (x2 - x1);
        v[i].y = (y2 - y1);
    }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            for (int k = 0; k < n; k++)
                for (int l = 0; l < n; l++) {
                    if (kol(v[i], v[j]) && kol(v[k], v[l]) && !used[i] && !used[j] && !used[k] && !used[l] && i != j && i != k && i != l && j != k && j != l && k != l) {
                        ans++;
                        //cout << i << ' ' << j << ' ' << k << ' ' << l << endl;
                        used[i] = used[j] = used[k] = used[l] = 1;
                    }
                }
    cout << ans;
    return 0;
}
```

Задача №4

Условие

На вход программе подаётся строка из строчных английских букв длиной < 1000 символов. Требуется выполнить "сжатие" этой строки по следующему принципу: подряд идущие повторяющиеся последовательности символов заменить на цепочки из трёх элементов: число - количество повторов, один экземпляр последовательности и точка.

В результате должна получиться строка наименьшей длины.

Входные данные: строка строчных английских букв.

Выходные данные: "сжатая" строка.

Исходные данные	Результат

aaaabbbbbc	4a.5b.c
abcdbcdbcdef	a3bcd.ef

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

int sz(int a) {
    int res = 0;
    if (!a) {
        return 1;
    }
    while (a) {
        res++;
        a /= 10;
    }
    return res;
}

string tos(int a) {
    string res = "";
    while (a) {
        res += '0' + a % 10;
        a /= 10;
    }
    reverse(res.begin(), res.end());
    return res;
}

int main() {
    string s, ans = "";
    cin >> s;
    while (!s.empty()) {
        string t = "";
        int mx = 0;
        for (int j = 1; j <= s.size(); j++) {
            int cnt = 0;
            string tmp = s.substr(0, j);
            for (int ind = 0; ind < s.size() - j + 1; ind += j) {
                if (s.substr(ind, j) == tmp)
                    cnt++;
                else
                    break;
            }
           // cout << cnt << ' ' << j << ' ' << mx << ' ' << t.size() << endl;
            if (cnt > mx) {
                t = tmp;
                mx = cnt;
            }
           // cout << t << ' ' << mx << endl;
        }
       // cout << t << ' ' << mx << endl;
        s.erase(0, mx * t.size());
        if (mx > 1) {
            ans += tos(mx) + t + '.';
        } else {
            ans += t;
        }
    }
    cout << ans;
    return 0;
}
```

Задача №5

Условие

Учитель информатики Игорь Петрович включил в программу подготовки 11-го класса изучение языка ассемблера.

Ассемблер - низкоуровневый язык программирования, каждая команда которого прямо соответствует машинной команде (инструкции процессора). Многие команды работают с регистрами - ячейками памяти, расположенными внутри процессора.

На текущий момент ученики Игоря Петровича прошли 3 команды: MOV, ADD, SUB.

Синтаксис этих команд таков:

- MOV <приёмник>, <источник> - копирует содержимое источника в приёмник.
- ADD <приёмник>, <источник> - прибавляет содержимое источника к приёмниу.
- SUB <приёмник>, <источник> - вычитает содержимое источника из приёмника.

Источником для каждой команды может быть числовая константа или название регистра, приёмником - только название регистра.

Доступные регистры - AX, BX, CX, DX. Ёмкость каждого регистра - 16 бит.

Значение регистров на момент начала выполнения программы не определено (то есть заранее неизвестно, может быть любым).

Для контроля успеваемости учеников Игорю Петровичу нужно оценивать понимание учениками "сложных" моментов в работе процессора. Требуется помочь ему составить программу, которая будет определять возникновение следующих ситуаций в коде на ассемблере: переполнение, переход через 0, использование неопределённых значений.

Переполнением будем называть ситуацию, когда в результате арифметической операции результат превысил 65535 или оказался меньше -32768. При этом старший разряд результата отбрасывается, а младшие остаются в регистре. Например, при прибавлении 65535 + 2 в регистре окажется 1, а при вычитании из -32768 1 в регистре окажется 32767. Отрицательные числа хранятся в виде дополнительного кода.

Переходом через 0 будем считать ситуацию, возникающую при изменении знака приёмника в результате сложения или вычитания.

Вход: фрагмент программы на ассемблере. Первая строка - целое положительное число, означающее количество команд и не превышающее 100. Каждая команда записана отдельной строкой, аргументы разделяются пробелами. Числовые константы лежат в диапазоне от -32768 до 65535.

Выход: 3 числа через пробел - количество случившихся переполнений, переходов через 0, использований неопределённых значений.

Исходные данные	Результат

3
MOV AX, 0
MOV BX, 5
SUB AX, BX
	0 1 0

4
 	
MOV AX, 1000
 	
MOV BX, 65000
 	
ADD AX, BX
 	
SUB CX, 128
	1 0 1

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

int toint(string s) {
    int res = 0;
    bool sig = 0;
    for (int i = 0; i < s.size(); i++) {
        if (s[i] == '-')
            sig = 1;
        else
            res = res * 10 + s[i] - '0';
    }
    if (sig)
        return -res;
    else
        return res;
}

int main() {
    short int reg[4];
    bool used[4] = {0};
    int n, ans1, ans2, ans3;
    string com, to, from, s;
    cin >> n;
    ans1 = ans2 = ans3 = 0;
    for (int i = 0; i < n; i++) {
        cin >> com >> to >> from;
        short int ind, ind1;
        int x;
        ind = (to[0] - 'A');
        if (com == "MOV") {
            if (from == "AX" || from == "BX" || from == "CX" || from == "DX") {
                ind1 = from[0] - 'A';
                if (!used[ind1]) {
                    ans3++;
                } else {
                    reg[ind] = reg[ind1];
                    used[ind] = 1;
                }
            } else {
                x = toint(from);
                if (x > 32767) {
                    ans1++;
                }
                reg[ind] = x;
                used[ind] = 1;
            }
        }


        if (com == "ADD") {
            if (from == "AX" || from == "BX" || from == "CX" || from == "DX") {
                ind1 = from[0] - 'A';
                if (!used[ind1]) {
                    ans3++;
                } else {
                    if (!used[ind]) {
                        ans3++;
                    } else {
                        x = reg[ind1];
                        if (int(reg[ind]) + x > 32767 || int(reg[ind]) + x < -32768)
                            ans1++;
                        else if (reg[ind] <= 0 && reg[ind] + x > 0 || reg[ind] >=0 && reg[ind] + x < 0)
                            ans2++;
                        reg[ind] += x;
                    }
                    //used[to] = 1;
                }
            } else {
                if (!used[ind]) {
                    ans3++;
                } else {
                    x = toint(from);
                    if (int(reg[ind]) + x > 32767 || int(reg[ind]) + x < -32768)
                        ans1++;
                    else if (reg[ind] <= 0 && reg[ind] + x > 0 || reg[ind] >=0 && reg[ind] + x < 0)
                        ans2++;
                    reg[ind] += x;
                    //used[to] = 1;
                }
            }
        }


        if (com == "SUB") {
            if (from == "AX" || from == "BX" || from == "CX" || from == "DX") {
                ind1 = from[0] - 'A';
                if (!used[ind1]) {
                    ans3++;
                } else {
                    if (!used[ind]) {
                        ans3++;
                    } else {
                        x = reg[ind1];
                        if (int(reg[ind]) - x > 32767 || int(reg[ind]) - x < -32768)
                            ans1++;
                        else if (reg[ind] <= 0 && reg[ind] - x > 0 || reg[ind] >=0 && reg[ind] - x < 0)
                            ans2++;
                        reg[ind] -= x;
                    }
                    //used[to] = 1;
                }
            } else {
                if (!used[ind]) {
                    ans3++;
```

```
                } else {
                    x = toint(from);
                    if (int(reg[ind]) - x > 32767 || int(reg[ind]) - x < -32768)
                        ans1++;
                    else if (reg[ind] <= 0 && reg[ind] - x > 0 || reg[ind] >=0 && reg[ind] - x < 0)
                        ans2++;
                    reg[ind] -= x;
                    //used[to] = 1;
                }
            }
        }
        //cout << reg[0] << ' ' << reg[1] << ' ' << reg[2] << ' ' << reg[3] << endl;
    }
    cout << ans1 << ' ' << ans2 << ' ' << ans3;
    return 0;
}
```

Задача №6

Условие

Из журнала отправки электронных писем по протоколу SMTP требуется определить адреса, отправка на которые не удалась.

Пример журнала:

mail1@gmail.com
250 bmstu.ru is pleased to meet you
250 cdp@bmstu.ru sender accepted
250 mail1@gmail.com accepting mail from a client address
354 Enter mail, end with "." on a line by itself
!!mail1@gmail.com 250
250 SMTP state reset

---
razin-ii 99@example.com
250 bmstu.ru is pleased to meet you
250 cdp@bmstu.ru sender accepted
555 syntax error in RCPT TO parameters
250 SMTP state reset

---
mail2@yandex.ru
250 bmstu.ru is pleased to meet you
250 cdp@bmstu.ru sender accepted
250 mail2@yandex.ru accepting mail from a client address
354 Enter mail, end with "." on a line by itself
!!mail2@yandex.ru 250
250 SMTP state reset

Каждая запись журнала состоит из нескольких строк. Записи разделяются последовательностью "----------" (10 дефисов). В первой строке записи указан почтовый адрес, на который направляется письмо. Далее идут строки, начинающиеся либо с трёхзначного кода SMTP, либо с группы из 8 пробелов, либо пустые строки. Если один из кодов, связанных с письмом, лежит в диапазоне от 500 до 599, значит, при отправке имела место ошибка. Так, второй адрес в примере содержит пробел, что привело к ошибке 555.

Входные данные: журнал отправки электронных писем. Журнал завершается строкой, состоящей из одного символа - точки.

Выходные данные: список адресов из журнала по порядку, попытка отправки писем на которые завершилась неудачно. Вывести по одному адресу в строке.

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

int toint(string s) {
    int res;
    if (!isdigit(s[0]) || !isdigit(s[1]) || !isdigit(s[2]))
        return -1;
    res = (s[0] - '0') * 100 + (s[1] - '0') + (s[2] - '0');
    return res;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int k = 0;
    string s, mail, p;
    set<string> adress;
    vector<string> ans;
    while (getline(cin, s)) {
        if (s == ".") break;
        if (s == "----------") {k++; p = s; continue;}
        int code = toint(s.substr(0, 3));
        if (code >= 500 && code < 600) {
            if (adress.count(mail) == 0) {
                adress.insert(mail);
                ans.push_back(mail);
            }
        }
        if (p == "----------" || !k) {
            mail = s;
            k++;
        }
        p = s;
    }
    for (int i = 0; i < ans.size(); i++)
        cout << ans[i] << endl;
    return 0;
}
```

Задача №7

Условие

Разреженной называется такая матрица, в которой количество ненулевых элементов существенно меньше количества нулевых. Разреженные матрицы можно хранить в форматах специального вида, например, только индексы ненулевых элементов и их значения.

Требуется написать программу, которая для разреженной матрицы большого размера определит суммы элементов в каждом столбце.

Входные данные: в первой строке три натуральных числа N, M и K, записанные через пробел. N - количество строк в разреженной целочисленной матрице, M - количество столбцов, K - количество ненулевых элементов. N и M не превышают 10^6 , K меньше N*M и не превышает 10^4. Далее по одному в строке указаны ненулевые элементы этой матрицы. Для представления каждого элемента через пробел записаны индексы его строки и столбца, а также значение. Индексы считаются с единицы, значения элементов по модулю не превышают 1000.

Выходные данные: M чисел по одному в строке - суммы элементов в соответствующих столбцах матрицы.

Исходные данные	Результат
4 3 1 1 1 10	10 0 0

Исходный код

```
#include <bits/stdc++.h>
#define f first
#define s secod

using namespace std;

typedef long long ll;
typedef long double ld;

ll sum[1000000];

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int n, m, k;
    cin >> n >> m >> k;
    for (int i = 0; i < k; i++) {
        int x, y;
        ll a;
        cin >> x >> y >> a;
        sum[y - 1] += a;
    }
    for (int i = 0; i < m; i++)
        cout << sum[i] << endl;
    return 0;
}
```