Киринцев Александр Сергеевич

Балл: 105

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 <iostream>
using namespace std;
int k,l,p;
int main(){
    cin >> k >> l >> p;
    int ans = 0;
    for(int i = 3; i <= min(k,p); i++){
        int a = min(k/i,(i != p ? l/(p-i) : 0)),tk = k,tl = l;
        tk -= a*i;
        tl -= a*(p-i);
        if(tk >= 3 && tl+tk == p){
            a++;
            ans = max(ans,a);
        }
        if(tk+tl == 0)
            ans = max(ans,a);
    }
    cout << ans;
    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 <iostream>
using namespace std;
//#define us unsigned short
typedef unsigned short us;
unsigned short int a,b;
unsigned short int st[16];
int main(){
    cin >> a >> b;

    for(short i = 0,x = 1; i <= 15; i++,x*=2)
        st[i] = x;
    if(a > b){
        int x = (a-b)/1024;
        int y = (a-b)-1024*x;
        int r = a;
        for(int i = 1; i <= 15;i++)
            if(us(a)*us(st[i]) >= b)
                r = r > a ? min(a*st[i],r) : a*st[i];

        int x1 = (r-b)/1024;
        int y1 = (r-b)-1024*x1;
        cout << min(x*2+y,4+x1*2+y1);
    }
    if (a < b){
        int r = a;
        for(int i = 1; i <= 15;i++)
            if(a*st[i] >= b)
                r = r > a ? min(a*st[i],r) : a*st[i];
        int x = (r-b)/1024;
        int y = (r-b)-1024*x;
        if(r-1024*x-y == b)
            cout << 4+x*2+y;
        else
            cout << -1;
    }

    if(a == b)
        cout << 0;

    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 параллелогораммов.

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 <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int n;
pair<int,int> a[100];
int t[100];
int main(){
    //freopen("00.txt","r",stdin);
    cin >> n;
    for(int i = 0; i < n; i++){
        int x1,y1,x2,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        x2-=x1;
        y2-=y1;
        a[i] = {x2,y2};
    }

    int j = 0;
    sort(a,a+n);
    if(a[1] == a[0])
        t[j]+=2;

    for(int i = 2; i < n; i++){
        if(a[i] == a[i-1])
            t[j]++;
        else
            t[++j]++;
    }
    sort(t,t+j+1,greater<int>());
    int q = 0;
    for(int i = 0; i < j+1; i++)
        for(int j = i+1;t[i] > 1; j++)
            while(t[j] > 1 && t[i] > 1){
                t[j]-=2;
                t[i]-=2;
                q++;
            }
    cout << q;
    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

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 <iostream>
#include <string>
#include <algorithm>
using namespace std;
int ls = -32768,lus = 65535; // limit short, limit unsigned short
int idx(char c) {
    if (c == 'A')
        return 0;
    if (c == 'B')
        return 1;
    if (c == 'C')
        return 2;
    if (c == 'D')
        return 3;
    return -1;
}

int main() {
    //freopen("00.txt","r",stdin);
    string s;
```

```
int n,f[4]= {0};
int a[4];
int q1=0,q2=0,q3=0;
cin >> n;
for(int i = 0; i <= n; i++) {
    getline(cin,s);
    if(s[0] == 'M') {
        f[idx(s[4])] = 1;
        string num;
        int j = 8;
        while('0' <= s[j] && s[j] <= '9')
            num+=s[j++];
        if(num.empty())
            a[idx(s[4])] = a[idx(s[8])];
        else
            a[idx(s[4])] = stoi(num);
    }
    if(s[0] == 'S'){
        string num;
        int j = 8;
        while('0' <= s[j] && s[j] <= '9')
            num+=s[j++];
        if(num.empty() && f[idx(s[4])]){
            if(f[idx(s[4])] == 0 || f[idx(s[8])] == 0){
                q2++;
                break;
            }
            if(a[idx(s[4])] - a[idx(s[8])] < 0 && a[idx(s[4])] >= 0){
                q1++;
            }
            a[idx(s[4])] -= a[idx(s[8])];
            if(a[idx(s[4])] < ls){
                a[idx(s[4])] = 32767-(ls-(a[idx(s[4])]));
                q3++;
            }
        }else{
            if(f[idx(s[4])] == 0){
                q2++;
                break;
            }
            if(a[idx(s[4])] - stoi(num) < 0 && a[idx(s[4])] > 0){
                q1++;
            }
            a[idx(s[4])] -= stoi(num);
            if(a[idx(s[4])] < ls){
                a[idx(s[4])] = 32767-(ls-(a[idx(s[4])]))+1;
                q3++;
            }
        }
    }
    if(s[0] == 'A'){
        string num;
        int j = 8;
        while('0' <= s[j] && s[j] <= '9')
            num+=s[j++];
        if(num.empty() && f[idx(s[4])]){
            if(f[idx(s[4])] == 0 || f[idx(s[8])] == 0){
                q2++;
                break;
            }
            if(a[idx(s[4])] + a[idx(s[8])] > 0 && a[idx(s[4])] < 0){
                q1++;
            }
            a[idx(s[4])] += a[idx(s[8])];
            if(a[idx(s[4])] > lus){
                a[idx(s[4])] = 0+(a[idx(s[4])]-lus);
                q3++;
            }
        }else{
            if(f[idx(s[4])] == 0){
                q2++;
                break;
            }
            if(a[idx(s[4])] + stoi(num) > 0 && a[idx(s[4])] < 0){
                q1++;
            }
            a[idx(s[4])] += stoi(num);
            if(a[idx(s[4])] > lus){
                a[idx(s[4])] = (a[idx(s[4])]-lus)-1;
                q3++;
            }
        }
    }
}

cout << q3 << " " << q1 << " " << q2;
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 <iostream>
#include <string>
#include <vector>
using namespace std;
vector<string> v;
int main(){
    //freopen("00.txt","r",stdin);
    string s;
    getline(cin,s);
    int f = 1,d = 1,n = 0;
    string add,num;
    while(s != "."){
        if(f == 1){
            add = s;
            f = 0;
        }
        if('0' <= s[0] && s[0] <= '9' && '0' <= s[1] && s[1] <= '9' && '0' <= s[2] && s[2] <= '9'){
            num = s[0];
            num += s[1];
            num += s[2];
            n = stoi(num);
        }
        if(d && 500 <= n && n <= 599){
```

```
            v.push_back(add);
            d = 0;
        }
        getline(cin,s);
        if (s=="----------"){
            getline(cin,s);
            f = 1;
            d = 1;
            n = 0;
        }
    }
    for(int i = 0; i < v.size(); i++)
        cout << v[i] << "\n";
    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 <iostream>
using namespace std;
int n,m,k,*a;
int main(){
    //freopen("00.txt","r",stdin);
    cin >> n >> m >> k;
    a = new int[1000000]{};
    for(int i = 0; i < k; i++){
        int x,y,z;
        cin >> x >> y >> z;
        a[y-1]+=z;
    }
    for(int i = 0; i < m; i++)
        cout << a[i] << "\n";
    return 0;
}
```