Викторов Антон Андреевич

Балл: 103

Задача №1

Условие

По просьбе учительницы математики Татьяны Григорьевны старшеклассники составили программу для проверки знаний пятиклассников по работе с натуральными дробями и смешанными числами. Вам требуется написать другую программу, которая для заданных примеров на сложение и вычитание смешанных чисел и дробей будет определять правильный ответ.

Входные данные: арифметическое выражение с двумя смешанными числами вида X Y/Z ? F G/H, где X - целая часть первого числа, Y - числитель дробной части первого числа, Z - знаменатель дробной части первого числа, F - целая часть второго числа, G - числитель дробной части второго числа, H - знаменатель дробной части второго числа. Между числами стоит знак операции “+” или “-” без кавычек. Целая, дробная части и знак операции разделены пробелами. Дробные части пробелов не содержат. Числа X, Y, Z, F, G, H не превышают 1000. У числа могут присутствовать целая и дробная части вместе или только одна из них.

Выходные данные: результат вычисления выражения в виде смешанного числа с дробной частью в виде сокращённой натуральной дроби, разделёнными одним пробелом, или целого числа.

Исходные данные	Результат
1 6/7 + 1/4	2 3/28
30/6 + 1	6

Исходный код

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

using namespace std;
#define int long long

pair<int, int> a, b;

int strtoint(string &s){
 int ans = 0, k = 1;
 for (int i = s.size() - 1; i >= 0; i--){
  ans += (s[i] - '0') * k;
  k *= 10;
 }
 return ans;
}

pair<int, int> getnumber(string s){
 int pos = s.find(' ');
 bool poc = 0;
 int z = 0, ch = 0, zn = 1;
 if (pos >= 0 && pos < s.size()){
  string zz = s.substr(0, pos);
  z = strtoint(zz);
  poc = 1;
 }
 else
  pos = -1;
 string chs, zns;
 int poss = s.find('/');
 if (poss >= 0 && poss < s.size()){
  poc = 1;
  for (int i = pos + 1; i < poss; i++)
   chs += s[i];
  for (int i = poss + 1; i < s.size(); i++)
   zns += s[i];
  ch = strtoint(chs);
  zn = strtoint(zns);
 }
 if (poc == 0){
  s += ' ';
  return getnumber(s);
 }
 return make_pair(zn * z + ch, zn);
}

int gcd(int a, int b){
 if (a > b)
  swap(a, b);
 while (a > 0){
  b %= a;
  swap(a, b);
 }
 return b;
}

signed main(){
 string s;
 getline(cin, s);
 char znak = '+';
 int posp = s.find('+');
 if (!(posp >= 0 && posp < s.size()))
  posp = s.find('-'), znak = '-';
 string aa, bb;
 for (int i = 0; i < posp - 1; i++)
  aa += s[i];
 for (int i = posp + 2; i < s.size(); i++)
  bb += s[i];
 a = getnumber(aa);
 b = getnumber(bb);
 pair<int, int> ans;
 //cerr << a.first << ' ' << a.second <<'\n' << b.first << ' ' << b.second << '\n';
 if (znak == '+')
  ans = make_pair(a.first * b.second + b.first * a.second, a.second * b.second);
 else
  ans = make_pair(a.first * b.second - b.first * a.second, a.second * b.second);
 if (ans.first * ans.second < 0){
  ans.first = abs(ans.first);
  ans.second = abs(ans.second);
  cout << '-';
 }
```

```
 int d = gcd(ans.first, ans.second);
 ans.first /= d;
 ans.second /= d;
 if (ans.first == 0){
  cout << 0;
  return 0;
 }
 //cerr << ans.first << ' ' << ans.second << '\n';
 if (ans.first >= ans.second)
  cout << ans.first / ans.second << ' ';
 ans.first %= ans.second;
 if (ans.first != 0)
  cout << ans.first << '/' << ans.second;
 return 0;
}
```

Задача №2

Условие

В квадратной матрице NxN требуется найти последовательность чисел, удовлетворяющую определённому соотношению. Числа должны располагаться под побочной диагональю или лежать на ней и при просмотре по спирали против часовой стрелки, начиная с правого верхнего угла, образовывать последовательность Фибоначчи.

Диагональ, идущая в квадратной матрице из левого нижнего угла в правый верхний, называется побочной. Числа Фибоначчи - элементы числовой последовательности, первые два числа которой равны 0 и 1, а каждое последующее - сумма двух предыдущих: 0, 1, 1, 2, 3, 5...

Входные данные: в первой строке задано единственное натуральное число N <= 100 - порядок матрицы. Далее идёт N строк по N целых чисел, разделённых одним или несколькими пробелами и описывающими элементы матрицы построчно. Каждый из элементов матрицы по модулю не превышает 10_6.

Выходные данные: найденные элементы последовательности Фибоначчи, удовлетворяющие условию, записанные в строку через пробел.

Исходные данные	Результат
4 0 1 2 3 4 5 **0 3** 7 8 **5 2** **1 1** 2 3	0 1 1 2 3 5

Примечание: жирным шрифтом выделены элементы, включаемые в результирующую последовательность.

Исходный код

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

using namespace std;
#define int long long

signed main(){
 int n;
 cin >> n;
 vector <vector <int> > g(n, vector <int> (n));
 for (int i = 0; i < n; i++)
  for (int j = 0; j < n; j++)
   cin >> g[i][j];
 int d = n, x = 0, y = n - 1;
 vector <int> v;
 while (d > 0){
  int xx = x, yy = y;
  for (int i = 0; i < d; i++){
   //cerr << 'd' << xx << ' ' << yy << '\n';
   v.push_back(g[xx][yy]);
   if (i != d - 1){
    xx++;
    yy--;
   }
  }

  yy++;
  for (int i = 1; i < d; i++){
   //cerr << "->" << xx << ' ' << yy << '\n';
   v.push_back(g[xx][yy]);
   if (i != d - 1)
   yy++;
  }
  xx--;
  for (int i = 2; i < d; i++){
   //cerr << xx << ' ' << yy << '\n';
   v.push_back(g[xx][yy]);
   if (i != d - 1)
   xx--;
  }
  x += 2;
  y--;
  d -= 3;
 }
 vector <int> fib(1, 0);
 for (int i = 0; i < v.size(); i++){
  //cerr << v[i] << ' ' << fib.back() << '\n';
  if (v[i] == fib.back()){
   cout << v[i] << ' ';
   if (fib.size() == 1)
    fib.push_back(1);
   else
    fib.push_back(fib[fib.size() - 2] + fib[fib.size() - 1]);
  }
 }
}
```

Задача №3

Условие

Для двух прямоугольников заданы координаты их вершин на плоскости, но так, что одной из вершин для каждого прямоугольника не хватает. Требуется восстановить координаты недостающих вершин и определить, пересекаются прямоугольники или нет. Под пересечением прямоугольников в рамках данной задачи понимать такое их расположение, при котором они имеют общие точки, не лежащие на периметре.

Входные данные: 2 строки по 6 вещественных чисел, записанных через пробел, в диапазоне от -1000 до 1000 и имеющих до 3-х знаков после точки. Каждая строка содержит координаты вершин x1, y1, x2, y2, x3, y3 соответствующего прямоугольника.

Выходные данные: NO, если фигуры не пересекаются, и YES, если пересекаются.

Исходные данные	Результат
1 0.1 3 0.1 1 1.1 7 11 11 11 7 15	NO
2 0 3 1 4 0 3 1 3 -1 5 1	YES

Исходный код

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

using namespace std;
#define int long long
#define ld long double

pair<pair<ld, ld>, pair<ld, ld> > pr(){
 ld x1, y1, x2, y2, x3, y3;
 cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
 pair<ld, ld> mi;
 mi.first = min(x1, min(x2, x3));
 mi.second = min(y1, min(y2, y3));
 pair<ld, ld> ma;
 ma.first = max(x1, max(x2, x3));
 ma.second = max(y1, max(y2, y3));
 return make_pair(mi, ma);
}

signed main(){
 pair<pair<ld, ld>, pair<ld, ld> > a, b;
 a = pr();
 b = pr();
 int counter = 0;
 if (a.first.first < b.second.first && a.first.second <b.second.second)
  counter++;
 swap(a, b);
 if (a.first.first < b.second.first && a.first.second <b.second.second)
  counter++;
 if (counter == 2)
  cout << "YES";
 else
  cout << "NO";
}
```

Задача №4

Условие

IP (Internet Protocol) - сетевой протокол, поддерживающий передачу пакетов между устройствами, расположенными в разных сетях (подсетях). В 4-й версии протокола (IPv4) адрес устройства состоит из четырёх октетов (8-разрядных неотрицательных чисел).

Подсетью называется (упрощённо) группа узлов, подключённых к одному сетевому коммутатору. Такие узлы могут направлять друг другу пакеты напрямую. В остальных случаях необходима маршрутизация пакета - передача его на маршрутизатор, который по специальной таблице определяет, в какую подсеть следует перенаправить пакет, чтобы он попал к адресату.

Подсети задаются с помощью битовых масок, которые совпадают с адресом по размеру и в двоичном представлении состоят из непрерывной последовательности единиц, дополненной нулями до длины адреса. Два узла расположены в одной подсети, если их адреса, логически умноженные на значение маски, совпадают.

Задана таблица маршрутизации сетевого устройства, работающего по протоколу IPv4. Требуется определить, по какому из маршрутов отправится пакет, направленный узлу с заданным адресом.

Входные данные: в первой строке - IP-адрес - строка из 4-х чисел от 0 до 255, разделённых точками, и количество маршрутов N (N <= 100). Далее N строк описывают таблицу маршрутизации. В каждой строке таблицы через пробел записаны: адрес сети, маска, приоритет. Маска записана в виде 4-х чисел от 0 до 255, разделённых точками, причём числа всегда заданы такие, что соответствующая им всем битовая последовательность не может содержать нулей, идущих левее, чем единицы.

IP-адрес, для которого требуется определить маршрут, не может содержать нулей в первом и последнем октетах. Адрес сети не может начинаться с нулевого октета. Маска должна содержать не менее одного единичного разряда.

Приоритет указывает на то, какой маршрут должен быть применён, если подходят несколько. В случае, если применимы несколько маршрутов с одинаковым приоритетом, выбирается первый из них.

Выходные данные: номер маршрута либо -1, если маршрута нет.

Исходные данные	Результат
172.16.10.10 3 172.16.10.0 255.255.255.0 10 10.0.0.0 255.192.0.0 5 172.16.8.0 255.255.252.0 1	1
172.16.9.10 3 172.16.10.0 255.255.255.0 10 10.0.0.0 255.192.0.0 5 172.16.8.0 255.255.252.0 1	3

Исходный код

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

using namespace std;
#define int long long
#define ld long double

vector <int> scan(){
 char q;
 vector <int> ans(4);
 cin >> ans[0] >> q >> ans[1] >> q >> ans[2] >> q >> ans[3];
 return ans;
}

signed main(){
 vector <int> ip = scan();
 int n, ans = -1, ansp = -1;
 cin >> n;
 for (int i = 0; i < n; i++){
  vector <int> tec = scan(), mask = scan();
  bool poc = 1;
  for (int i = 0; i < 4; i++){
```

```
   //cerr << mask[i] << '&' << ip[i] <<'(' << (mask[i] & ip[i]) << ')' << '=' << tec[i] << "?\n";
   if ((tec[i]) != (mask[i] & ip[i]))
    poc = 0;
  }
  int p;
  cin >> p;
  if (poc && p > ansp)
   ans = i + 1, ansp = p;
 }
 cout << ans;
}
```

Задача №5

Условие

Двоичным деревом называется иерархическая структура данных, в которой каждый узел имеет не более двух потомков.

Двоичное дерево поиска - разновидность двоичного дерева, у которого оба поддерева (левое и правое) являются двоичными деревьями поиска; все узлы левого поддерева произвольного узла A меньше, чем значение самого узла A; все узлы правого поддерева произвольного узла A больше либо равны значению узла A.

При прямом обходе дерева сначала обрабатывается ключ (значение) корня, затем - ключи левого и правого поддеревьев. При обратном обходе сначала обрабатываются ключи левого поддерева, затем правого, затем - корня.

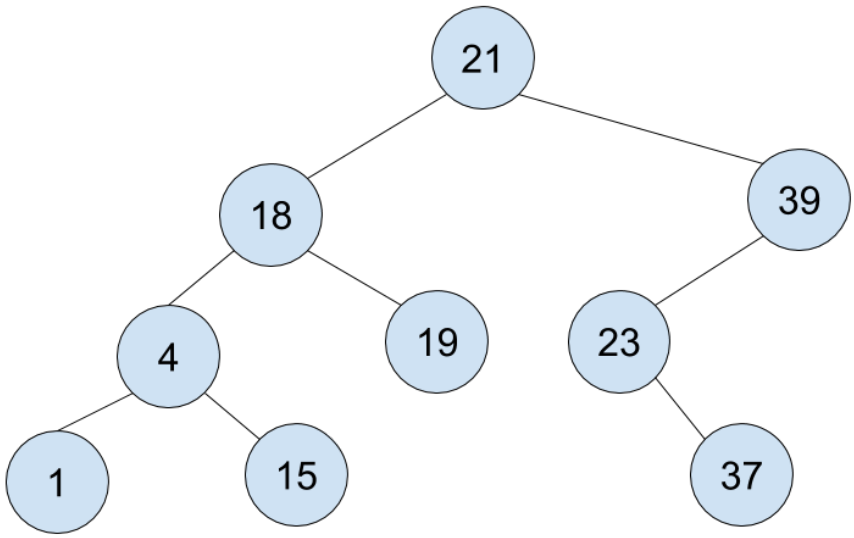
По заданным ключам вершин, полученным при прямом обходе, требуется вывести ключи того же дерева, получаемые при обратном обходе.

Входные данные: ключи всех вершин двоичного дерева поиска в порядке прямого обхода. Каждый ключ - натуральное число, не превышающее 10^5. Каждое значение задано в отдельной строке. Последняя строка состоит из одного символа - точки.

Выходные данные: ключи дерева, соответствующие обратному порядку обхода, по одному в строке.

Пример

Исходные данные и результат соответствуют дереву, изображённому на рисунке:



Исходные данные	Результат

Исходный код

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

using namespace std;
#define int long long
#define ld long double

struct node{
 node *l, *r, *p;
 int x;
};

void add(int x, node* &t){
 if (x < t->x){
  t->l = new node;
```

21	1
18	15
4	4
1	19
15	18
19	37
39	23
23	39
37	21
.	

```
  }
  //cerr << t->x << '\n';
  if (t->p == NULL || t->p->x > x){
   t->r = new node;
   t->r->p = t;
   t->r->l = NULL;
   t->r->r = NULL;
   t->r->x = x;
   t = t->r;
   return;
  }
  else{
   t = t->p;
   add(x, t);
  }
 }
}

int strtoint(string &s){
 int ans = 0, k = 1;
 for (int i = s.size() - 1; i >= 0; i--){
  ans += (s[i] - '0') * k;
  k *= 10;
 }
 return ans;
}

void obh(node *t){
 if (t->l != NULL)
  obh(t->l);
 if (t->r != NULL)
  obh(t->r);
}

void ans(node *t){
 if (t->l != NULL){
  //cerr << t->x << " - parent of " << t->l->x << '\n';
  ans(t->l);
 }
 if (t->r != NULL){
  //cerr << t->x << " - parent of " << t->r->x << '\n';
  ans(t->r);
 }
 cout << t->x << "\n";
}

signed main(){
 node *t = new node;
 t->l = NULL;
 t->r = NULL;
 t->p = NULL;
 string s;
 int x;
 bool poc = 1;
 while (cin >> s){
  if (s == ".")
   break;
  x = strtoint(s);
  if (poc == 1){
   poc = 0;
   t->x = x;
  }
  else
   add(x, t);

 }
 while (t->p != NULL)
  t = t->p;
 ans(t);
}
```

```
  t->l->p = t;
  t->l->l = NULL;
  t->l->r = NULL;
  t->l->x = x;
  t = t->l;
  return;
 }
 else{
  bool q = 1;
  if (t->p != NULL && t->p->r == t){
   node *tt = t;
   while (tt->p != NULL){
    if (tt->p->r == tt)
     q = 1;
    else
     q = 0;
    tt = tt->p;
    if (tt->x <= x && !q){
     t = tt;
     break;
    }
   }
```

Задача №6

Условие

В связи с внезапно возникшей вспышкой неизвестного заболевания министерство образования Тридевятого царства решило перевести студентов всех вузов на домашнюю форму обучения. Кроме того, очередной приём в вузы также запланировали провести дистанционно. Таким образом, абитуриенты теперь будут подавать документы на поступление через Тридевятую компьютерную сеть. Предполагается, что все смогут подать заявления на интересующие их специальности, не выходя из дома, а подведение итогов конкурса произойдёт автоматически, когда завершится подача заявлений.

Каждый вуз подводит итоги конкурса независимо друг от друга. Для этого в вузе составляются **конкурсные списки** абитуриентов по каждой специальности, ранжированные по убыванию суммы их баллов по трём предметам. В случае равенства суммы баллов производится дополнительное ранжирование по убыванию баллов за 1-й предмет, затем за 2-й, затем за 3-й. Если выявляется несколько абитуриентов с абсолютно одинаковыми баллами, проходящих на специальность, где свободных мест осталось меньше их количества - принимаются все такие абитуриенты.

Кроме того, в каждой записи указывается, какой приоритет для абитуриента имеет эта специальность.

Согласно порядку приёма, абитуриент может подать заявления не более, чем на 3 специальности в один вуз.

Требуется написать программу, которая по заданным конкурсным спискам определит списки зачисленных на каждую специальность.

Входные данные: в первой строке - натуральное число N <= 50 - количество специальностей в вузе. Далее - N описаний конкурсных списков по каждой специальности.

Описание конкурсного списка начинается с заголовка - строки, где указан шифр или название специальности - строка из цифр, латинских букв, знаков подчёркивания, точек и запятых, длина которой не превышает 50 символов, и через пробел - количество мест для приёма. Количество мест - натуральное число, не превышающее 200. Далее идут строки, в каждой из которых указана информация об очередном абитуриенте. Строка с описанием абитуриента состоит из его имени - слова, записанного латинскими буквами, длина которого не превышает 20; затем через пробел 3 оценки - натуральных числа, не превышающих 100; затем натуральное число от 1 до 3 - приоритет данной специальности согласно предпочтению абитуриента.

Описание каждого конкурсного списка заканчивается пустой строкой.

Гарантируется, что у каждого абитуриента приоритеты не повторяются и номер приоритета не превышает количество выбранных им специальностей. Баллы каждого абитуриента одинаковы на всех специальностях.

Выходные данные: списки поступивших (зачисленных) на каждую специальность. Порядок специальностей должен соответствовать исходному.

В первой строке каждого списка указывается её название и через пробел - количество зачисленных; далее в каждой строке - имя абитуриента. Имена абитуриентов следует отсортировать по возрастанию в алфавитном порядке.

Исходные данные	Результат
2 PrikladnayaMatematika 2 Alena 99 99 100 1 Pavel 99 90 90 1 Ilya 100 89 80 1 IVT 3 Vasya 99 99 98 1 Ilya 100 89 80 2	PrikladnayaMatematika 2 Alena Pavel IVT 2 Ilya Vasya

Исходный код

```
//#include "stdafx.h"
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <cmath>
#include <map>
using namespace std;

struct stud{
 vector <int> b;
 int p;
 string name;
};

int strtoint(string &s){
 int ans = 0, k = 1;
 for (int i = s.size() - 1; i >= 0; i--){
  ans += (s[i] - '0') * k;
  k *= 10;
 }
 return ans;
}

struct spec{
 string name;
 stud gr;
 int k;
 vector <stud> a;
};

bool comp(stud a, stud b){
 int suma = a.b[0] + a.b[1] + a.b[2], sumb = b.b[0] + b.b[1] + b.b[2];
 if (suma < sumb) return 0;
 if (sumb < suma) return 1;
 for (int i = 0; i < 3; i++){
  cerr <<a.b[i] << ' ' << b.b[i] << '\n';
  if (a.b[i] < b.b[i])
   return 0;
 }
 return 1;
};

bool equal(stud a, stud b){
 return ((a.b[0] == b.b[0]) && (a.b[1] == b.b[1]) && (a.b[2] == b.b[2]));
}

signed main(){
 stud genius;
 genius.name ="genius";
 genius.b.assign(3, 1000);
 genius.p = -1;

 int n;
 cin >> n;
 map<string, bool> m;
 vector <spec> v(n);
 for (int i = 0; i < n; i++){
  cin >> v[i].name >> v[i].k;
  v[i].gr = genius;
  string s;
  //cin >> s >> s >> s;
  getline(cin, s);
  while (getline(cin, s)){
   if (s == "")
    break;
   stud q;
   q.name = s.substr(0, s.find(' '));
   s.erase(0, s.find(' ') + 1);
   q.b.resize(3, 0);
   for (int j = 0; j < 3; j++){
    q.b[j] = strtoint(s.substr(0, s.find(' '))));
    s.erase(0, s.find(' ') + 1);
   }
   q.p = strtoint(s);
   v[i].a.push_back(q);
  }
 }
 vector <vector <string> > ans(n);
 for (int tecp = 1; tecp <= 3; tecp++){
```

```
  for (int w = 0; w < 750; w++){
   for (int i = 0; i < n; i++){
    vector <stud> a;
    for (int j = 0; j < (int)(v[i].a.size()); j++){
     if (!m[v[i].a[j].name])
      a.push_back(v[i].a[j]);
    }
    sort(a.begin(), a.end(), comp);
    int j = 0;
    for (j = 0; j < a.size() && v[i].k > 0; j++){
     v[i].gr = a[j];
     m[a[j].name] = 1;
     ans[i].push_back(a[j].name);
     v[i].k--;
    }
    while (j != (int)(a.size())){
     if (equal (a[j], v[i].gr)){
      v[i].gr = a[j];
      m[a[j].name] = 1;
      ans[i].push_back(a[j].name);
      v[i].k--;
     }
     j++;
    }
    v[i].a = a;
   }
  }
 }
 for (int i = 0; i < n; i++){
  cout << v[i].name << ' ' << ans[i].size() << '\n';
  sort(ans[i].begin(), ans[i].end());
  for (int j = 0; j < ans[i].size(); j++)
   cout << ans[i][j] << '\n';
 }
}
```

Задача №7

Условие

Требуется решить задачу №6 с тем отличием, что наборы экзаменов на разные программы различаются, а абитуриенты имеют более трёх оценок.

Для подведения итогов конкурса в вузе составляются **конкурсные списки** абитуриентов по каждой специальности, ранжированные по убыванию суммы их баллов по трём предметам. В случае равенства суммы баллов производится дополнительное ранжирование по убыванию баллов за 1-й предмет, затем за 2-й, затем за 3-й. Если выявляется несколько абитуриентов с абсолютно одинаковыми баллами, проходящих на специальность, где свободных мест осталось меньше их количества - принимаются все такие абитуриенты.

Кроме того, в каждой записи указывается, какой приоритет для абитуриента имеет эта специальность.

Согласно порядку приёма, абитуриент может подать заявления не более, чем на 3 специальности в один вуз.

Требуется написать программу, которая по заданным конкурсным спискам определит списки зачисленных на каждую специальность.

Входные данные: в первой строке - натуральное число N <= 50 - количество специальностей в вузе. Далее - N описаний конкурсных списков по каждой специальности.

Описание конкурсного списка начинается с заголовка - строки, где указан шифр или название специальности - строка из цифр, латинских букв, знаков подчёркивания, точек и запятых, длина которой не превышает 50 символов, количество мест для приёма и шифры трёх предметов, закодированные заглавными латинскими буквами. Элементы строки разделены пробелами. Количество мест - натуральное число, не превышающее 200. Далее идут строки, в каждой из которых указана информация об очередном абитуриенте. Строка с описанием абитуриента состоит из его имени - слова, записанного латинскими буквами, длина которого не превышает 20; затем через пробел 3 оценки - натуральных числа, не превышающих 100; затем натуральное число от 1 до 3 - приоритет данной специальности согласно предпочтению абитуриента.

Описание каждого конкурсного списка заканчивается пустой строкой.

Гарантируется, что у каждого абитуриента приоритеты не повторяются и номер приоритета не превышает количество выбранных им специальностей. Баллы каждого абитуриента по одному и тому же предмету одинаковы на всех специальностях.

Выходные данные: списки поступивших (зачисленных) на каждую специальность. Порядок специальностей должен соответствовать исходному.

В первой строке каждого списка указывается её название и через пробел - количество зачисленных; далее в каждой строке - имя абитуриента. Имена абитуриентов следует отсортировать по возрастанию в алфавитном порядке.

Исходные данные	Результат

Исходный код

```
//#include "stdafx.h"
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <cmath>
#include <map>
using namespace std;

struct stud{
 vector <int> b;
 int p;
 string name;
};

int strtoint(string &s){
 int ans = 0, k = 1;
 for (int i = s.size() - 1; i >= 0; i--){
  ans += (s[i] - '0') * k;
  k *= 10;
 }
 return ans;
}

struct spec{
 string name;
 stud gr;
 int k;
 vector <stud> a;
};
```

3	PrikladnayaMatematika 2
PrikladnayaMatematika 2 F M R	Alena
Alena 99 99 100 1	Pavel
Pavel 99 90 90 1	IVT 3
Ilya 100 90 80 1	Dima
	Nikita
IVT 3 I M R	Vasya
Alena 99 99 100 2	InformSistemy 1
Vasya 99 99 98 1	Ilya
Pavel 95 90 90 2	
Dima 95 90 90 1	
Nikita 90 90 81 1	
Ilya 90 90 80 2	
InformSistemy 2 I M R	
Vasya 99 99 98 2	
Ilya 90 90 80 3	
Nikita 90 90 81 2	

```
    q.b[j] = strtoint(s.substr(0, s.find(' '))); 
    s.erase(0, s.find(' ') + 1);
   }
   q.p = strtoint(s);
   v[i].a.push_back(q);
  }
 }
 vector <vector <string> > ans(n);
 for (int tecp = 1; tecp <= 3; tecp++){
  for (int w = 0; w < 750; w++){
   for (int i = 0; i < n; i++){
    vector <stud> a;
    for (int j = 0; j < (int)(v[i].a.size()); j++){
     if (!m[v[i].a[j].name])
      a.push_back(v[i].a[j]);
    }
    sort(a.begin(), a.end(), comp);
    int j = 0;
    for (j = 0; j < a.size() && v[i].k > 0; j++){
     v[i].gr = a[j];
     m[a[j].name] = 1;
     ans[i].push_back(a[j].name);
     v[i].k--;
    }
    while (j != (int)(a.size())){
     if (equal (a[j], v[i].gr)){
      v[i].gr = a[j];
      m[a[j].name] = 1;
      ans[i].push_back(a[j].name);
      v[i].k--;
     }
     j++;
    }
    v[i].a = a;
   }
  }
 }
 for (int i = 0; i < n; i++){
  cout << v[i].name << ' ' << ans[i].size() << '\n';
  sort(ans[i].begin(), ans[i].end());
  for (int j = 0; j < ans[i].size(); j++)
   cout << ans[i][j] << '\n';
 }
}
```

```
bool comp(stud a, stud b){
 int suma = a.b[0] + a.b[1] + a.b[2], sumb = b.b[0] + b.b[1] + b.b[2];
 if (suma < sumb) return 0;
 if (sumb < suma) return 1;
 for (int i = 0; i < 3; i++){
  cerr <<a.b[i] << ' ' << b.b[i] << '\n';
  if (a.b[i] < b.b[i])
   return 0;
 }
 return 1;
};

bool equal(stud a, stud b){
 return ((a.b[0] == b.b[0]) && (a.b[1] == b.b[1]) && (a.b[2] == b.b[2]));
}

signed main(){
 stud genius;
 genius.name ="genius";
 genius.b.assign(3, 1000);
 genius.p = -1;

 int n;
 cin >> n;
 map<string, bool> m;
 vector <spec> v(n);
 for (int i = 0; i < n; i++){
  cin >> v[i].name >> v[i].k;
  v[i].gr = genius;
  string s;
  //cin >> s >> s >> s;
  getline(cin, s);
  while (getline(cin, s)){
   if (s == "")
    break;
   stud q;
   q.name = s.substr(0, s.find(' '));
   s.erase(0, s.find(' ') + 1);
   q.b.resize(3, 0);
   for (int j = 0; j < 3; j++){
```