Заключительный этап академического соревнования Олимпиады школьников «Шаг в будущее» по программированию «Профессор Лебедев» (общеобразовательный предмет информатика), весна 2021 год

11 класс

Вариант 1

Задача 1 (8 баллов)

Условие

Задано натуральное число N, не превышающее 10^8. Требуется определить, сколько чётных цифр входит в его запись в троичной системе счисления.

Входные данные: число N, записанное в десятичной системе счисления.

Выходные данные: количество чётных цифр, входящих в запись числа N в троичной системе счисления.

Пример

Входные данные	Выходные данные	Пояснение
4	0	$4_{10}=11_3$
15	2	$15_{10}=120_3$

Проверочные тесты

Входные данные	Ожидаемый результат
4	0
15	2
2	1
8	2
9876543	12

Пример решения

```
#include<stdio.h>
#include<bits/stdc++.h>

using namespace std;

int main()
{
    int x = 0;
    cin >> x;
    int n [10000];
    int i = 0;
    int ans=0;
    while (x != 0)
    {
```

```
        n[i]= x % 3;
        x/=3;
        i++;
    }
    for (int j = i - 1; j >= 0; j--)
    {
        if (n[j] % 2 == 0)
        {
            ans++;
        }
    }
    cout << ans;
    return 0;
}
```

Задача 2 (12 баллов)
Условие
Исключающим "или" (XOR) называется булева функция, а также логическая и битовая операция от двух аргументов, результат которой истинен тогда и только тогда, когда один из аргументов истинен, а второй - ложен.

Циклический побитовый сдвиг вправо - операция, при которой младший разряд переносится в начало числа и становится старшим, а все остальные сдвигаются вправо на одну позицию.

К двум 16-битовым числам A и B, записанным в 16-ричной системе счисления, была применена операция исключающего "или", а затем к результату - операция побитового циклического сдвига вправо на K разрядов. Одно из двух исходных чисел было забыто, требуется его восстановить.

Входные данные: в строку через пробел записаны числа A, K и результат X. Числа A и X заданы в 16-ричной системе счисления, K - в десятичной.

Выходные данные: число B.

Пример:

Входные данные	Выходные данные
1A2B 4 4E5D	FFFF
AB00 1 5C9A	1234

Проверочные тесты

Входные данные	Ожидаемый результат
1A2B 4 4E5D	FFFF
AB00 1 5C9A	1234
FFFF 1 EEEE	2222
7777 8 FFFF	8888
FEDC 3 1D9D	1234

Пример решения

```
def normal_hex(a):
    add = (4 - len(a) % 4) % 4
    return '0'*add + a

a, k, x = input().split()
a, k, x = bin(int(a, 16)), int(k), bin(int(x, 16))
cur1 = str(a)[2::]
c = str(x)[2::]
c = normal_hex(c)
cur1 = normal_hex(cur1)
k %= len(a)
c = c[k::] + c[:k:]
#print(cur1)
#print(c)
answ = ''
count = 0
while count < len(c):
    cur = c[count]
    try:
        temp = cur1[count]
        if cur == temp:
            answ = answ + '0'
        else:
            answ = answ + '1'
    except IndexError:
        answ = answ + cur
    count += 1
#print(answ)
result = str(hex(int(answ, 2)))[2::]
result = result.upper()
print(result)
```

Задача 3 (16 баллов)

Условие

В связи с приближающимся завершением эпидемии коронавируса правительство города Энска приняло решение отправить на южные морские курорты отличников школ города - N отдыхающих. В авиакомпаниях для перевозки имеются K типов самолётов с различным количеством пассажирских мест (P_1, P_2,...,P_k). Необходимо решить логистическую задачу: зарезервировать необходимое количество рейсов (R_1, R_2,...,R_k).

Суммарное количество посадочных мест не может быть меньше, чем количество пассажиров, а общее количество незанятых пассажирских мест (V) должно быть минимальным. Если будет обнаружено несколько вариантов, одинаковых по количеству незанятых мест, то следует выбрать вариант с меньшим общим количеством рейсов самолётов. Если и таких вариантов будет несколько, то нужно вывести любой из них.

Входные данные: в первой строке через пробел записаны 2 натуральных числа N и K, во второй - K натуральных чисел P_1 P_2 … P_k. N не превышает 10000, K не превышает 20, любое P_i меньше 500.

Выходные данные: в первой строке вывести число V, во второй через пробел вывести R_1 R_2 … R_k.

Порядок количеств рейсов должен соответствовать исходному порядку типов самолётов.

Пример

Входные данные	Выходные данные
1000 3 300 220 150	30 0 4 1
80 4 100 50 40 30	0 0 1 0 1 (вариант - 0 0 2 0)

Проверочные тесты

Входные данные	Ожидаемый результат
2000 3 600 440 300	60 0 4 1
80 4 100 50 40 25	0 0 0 2 0
100 1 100	0 1
1000 3 300 200 100	0 3 0 1
4960 4 350 300 200 150	40 10 5 0 0

Пример решения

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int con = 100000;

signed main() {
    //freopen("input.txt", "r", stdin);
    //freopen("output.txt", "w", stdout);
    int n, k;
    cin >> n >> k;
    int ma = n;
    vector<int> p(k);
    for (int i = 0; i < k; i++) {
        cin >> p[i];
    }
    vector<pair<int, int> > d(con);
    d[0] = {0, 0};
    for (int i = 1; i < con; i++) {
        d[i] = {-1, 1000000000};
        for (int j = 0; j < k; j++) {
            if (i >= p[j]) {
                if (d[i].second > d[i - p[j]].second + 1) {
                    d[i] = {i - p[j], d[i - p[j]].second + 1};
```

```
                }
            }
        }
    }
    for (int i = n; i < con; i++) {
        if (d[i].first != -1) {
            cout << i - n << "\n";
            vector<int> ans(k);
            int del;
            while (i != 0) {
                del = i - d[i].first;
                i = d[i].first;
                for (int j = 0; j < k; j++) {
                    if (del == p[j]) {
                        ans[j]++;
                        j = k + 1;
                    }
                }
            }
            for (int j = 0; j < k; j++) {
                cout << ans[j] << " ";
            }
            return 0;
        }
    }
}
```

Задача 4 (20 баллов)
Условие
Миссия космолёта “Юрий Гагарин” определена - посещение трех звёздных систем в глубоком космосе. Эти звёзды находятся на Земном небосводе в созвездии Орион:

- звезда Беллатрикс (250 световых лет от Земли)
- звезда Бетельгейзе (643 световых года от Земли)
- парная звёздная система Ригель (773 световых года от Земли)

Маршрут путешествия должен предусматривать посещение их в порядке удаления от Земли. Однако гиперсветовой прыжок не может пока быть выполнен с приемлемой точностью на расстояние 20 и более световых лет. Спланированная трасса полёта должна состоять из серии прыжков, каждый прыжок менее безопасного расстояния. Вторым ограничением является то, что промежуточные точки трассы должны находиться в окрестности массивного тела, т.е. звезды. Такие звёзды в навигации называются контрольными пунктами.

Необходимо учитывать, что управление выходом из гиперпространства невозможно без существенного искривления его в точке назначения гравитационным воздействием звезды контрольного пункта. Эта особенность налагает дополнительное ограничение на каждый выполняемый гиперпрыжок: траектория прыжка не должна проходить меньше одного светового года от звёздной системы, не являющейся контрольным пунктом.

Задача : построить трассу в трёхмерном пространстве с минимально возможным количеством гиперпрыжков, при соблюдении указанных выше ограничений. Если таких трасс несколько, то следует выбрать ту, в которой суммарная длина трассы меньше.

ВХОДНЫЕ ПАРАМЕТРЫ:

Первая строка : натуральное число N - количество звезд, внесённых в лоцию (от 3 до 100).

Далее N строк, каждая содержит три целых числа, записанных через пробел: координаты X Y Z, заданные в световых годах. Все значения координат не превышают 10000.

Точка старта (0 0 0) - звезда Солнце - не указана в лоции.

Три посещаемые звёздные системы находятся в лоции в порядке ожидаемого посещения, в строках с номерами 1, 2 и 3.

ВЫХОДНЫЕ ПАРАМЕТРЫ:

Первая строка: через пробел должны быть указаны количество прыжков и длина трассы с точностью до третьего знака после запятой.

Вторая строка: номера звёзд-”контрольных пунктов” и звёзд-”целей миссии” в порядке следования звездолёта, записанные через пробел.

Примечание: для отработки алгоритма построения маршрута космолёта в лоцию (комплект навигационной информации для судовождения) могут включаться различные произвольные координаты, не соответствующие известным звёздам.

В рамках задачи считать, что движение космолёта между контрольными точками происходит прямолинейно.

Пример

Входные данные	Выходные данные
3 10 10 10 20 20 20 30 30 30	3 51,962 1 2 3
5 20 0 0 10 0 0 40 0 0 30 0 0 50 0 0	4 40,000 2 1 4 3

Проверочные тесты

Входные данные	Ожидаемый результат
3 10 10 10 20 20 20 30 30 30	3 51.962 1 2 3

5 20 0 0 10 0 0 40 0 0 30 0 0 50 0 0	4 40.000 2 1 4 3
10 10 10 10 50 50 50 70 70 70 20 20 20 30 30 30 40 40 40 60 60 60 80 80 80 90 90 90 95 95 95	7 121.244 1 4 5 6 2 7 3
10 1 1 1 5 5 5 7 7 7 2 2 2 3 3 3 4 4 4 6 6 6 8 8 8 9 9 9 10 10 10	7 12.124 1 4 5 6 2 7 3
10 1 1 1 5 5 5 7 7 7 2 0 2 3 3 1 2 4 4 8 6 6 8 9 8 9 9 8 10 10 10	3 12.124 1 2 3

Пример решения

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <set>
#include <utility>
#include <algorithm>
#include <cmath>
```

```
#include <queue>

using namespace std;

double inf = 1e9;
long n;
vector<long> x;
vector<long> y;
vector<long> z;

long sq(long n) { return n * n; }

double len2(long v, long u) {
     return sq(x[v] - x[u]) + sq(y[v] - y[u]) + sq(z[v] - z[u]);
}

long vecmul(long a, long o, long b) {
     return (x[a] - x[o]) * (x[b] - x[o]) + (y[a] - y[o]) * (y[b] - y[o])
+ (z[a] - z[o]) * (z[b] - z[o]);
}

bool pathClear(long v, long u) {
     for (long k = 0; k <= n; ++k) {
          if (v == k || k == u || v == u || vecmul(k, v, u) < 0 ||
vecmul(k, u, v) < 0) {
               continue;
          }

          double a = sqrt(len2(v, u));
          double b = sqrt(len2(k, u));
          double c = sqrt(len2(v, k));
          double p = (a + b + c) / 2;
          double s = sqrt(p * (p - a) * (p - b) * (p - c));

          if (2 * s / a < 1) {
               return false;
          }
     }

     return true;
}

pair< vector<long>, vector<long> > findPath(long s) {
     vector<long> d(n + 1, 1e5);
     vector<long> p(n + 1, -1);
     d[s] = 0;
     p[s] = 0;
     queue<long> q;
     vector<bool> used(n + 1, false);

     for(q.push(s); !q.empty(); q.pop()) {
          long v = q.front();
```

```
            if (used[v]) {
                continue;
            }

            used[v] = true;

            for (long u = 0; u < n + 1; ++u) {
                if (used[u] || len2(v, u) > 400 || !pathClear(v, u)) {
                    continue;
                }

                d[u] = d[v] + 1;
                p[u] = v;
                q.push(u);
            }
        }

        return pair< vector<long>, vector<long> >(d, p);
}

int main() {
        cout << fixed;
        cout.precision(3);

        cin >> n;
        x.resize(n + 1, 0);
        y.resize(n + 1, 0);
        z.resize(n + 1, 0);

        x[0] = 0;
        y[0] = 0;
        z[0] = 0;

        for (long i = 1; i <= n; ++i) {
            cin >> x[i] >> y[i] >> z[i];
        }

        auto p0 = findPath(0);
        auto p1 = findPath(1);
        auto p2 = findPath(2);
        auto p3 = findPath(3);

        vector< vector<long> > d;
        d.push_back(p0.first);
        d.push_back(p1.first);
        d.push_back(p2.first);
        d.push_back(p3.first);

        vector< vector<long> > p;
        p.push_back(p0.second);
        p.push_back(p1.second);
        p.push_back(p2.second);
        p.push_back(p3.second);
```

```
	long curMin = 1e5;
	vector<long> ans;
	double len = 0;
	for (long a = 1; a <= 3; ++a) {
		for (long b = 1; b <= 3; ++b) {
			for (long c = 1; c <= 3; ++c) {
				if (a == b || b == c || c == a) {
					continue;
				}
				long sum = d[0][a] + d[a][b] + d[b][c];
				// cout << d[0][a] <<  ' '  << d[a][b] <<  ' '  <<
d[b][c] <<  '\n';
				if (sum < curMin) {
					curMin = sum;
					ans.clear();
					len = 0;
					long v = c;

					while (v != b) {
						ans.push_back(v);
						long u = p[b][v];
						len += sqrt(len2(v, u));
						v = u;
					}

					while (v != a) {
						ans.push_back(v);
						long u = p[a][v];
						len += sqrt(len2(v, u));
						v = u;
					}

					while (v != 0) {
						ans.push_back(v);
						long u = p[0][v];
						len += sqrt(len2(v, u));
						v = u;
					}
				}
			}
		}
	}

	reverse(ans.begin(), ans.end());

	cout << curMin << ' ' << len << '\n';
	for (auto a : ans) {
		cout << a << ' ';
	}

	// system("pause");
	return 0;
```

```
}
```

Задача 5 (20 баллов)
Условие
Стек - тип данных, представляющий собой список элементов, организованных по принципу "последним пришёл - первым ушёл". В большинстве архитектур компьютерных систем стек применяется для организации вызова подпрограмм, поскольку лучше всего подходит для обеспечения работы рекурсии.

Фирма Хардсофт занялась разработкой инновационного процессора с переменной длиной кодов команд. Одновременно одно из её подразделений начало работу над компилятором для нового процессора. Им удалось создать алгоритм предсказания запуска той или иной подпрограммы на основе исходного кода, и теперь разработчики хотят на основе полученной информации научиться определять оптимальный размер стека.

Вам требуется помочь им составить программу для получения всех возможных цепочек вызовов подпрограмм на основе результатов работы их алгоритма.

Входные данные: в первой строке записано натуральное число N, не превышающее 10 - максимальная глубина вложенности вызовов подпрограмм. Начиная со второй строки, идёт протокол работы алгоритма предсказания запуска подпрограмм, который заканчивается строкой, состоящей из одной точки. Пример протокола:
PROC A
ENDPROC
PROC B
CALL A
ENDPROC
PROC MAIN
CALL B
ENDPROC
.

Описание подпрограммы в нём начинается со слова PROC, после которого через идёт имя из латинских букв, длина которого не превышает 20 символов.

Внутри подпрограммы могут указываться вызовы других подпрограмм с помощью слова CALL и через пробел - имени вызываемой подпрограммы.

Описание подпрограммы завершается словом ENDPROC.

Описания подпрограмм не могут пересекаться и быть вложенными. Первой всегда выполняется подпрограмма MAIN и она всегда есть в протоколе.

Выходные данные: в первой строке - количество выявленных цепочек, в последующих строках - цепочки вызовов, соответствующие исходному протоколу. Каждая цепочка должна быть выведена с новой строки, а имена подпрограмм в ней должны быть приведены в том порядке, в котором происходят их вызовы, и записаны через пробел. Имя MAIN выводить не нужно, работа главной подпрограммы в подсчёт вложенности вызовов не входит.

Цепочки должны быть выведены в том порядке, в котором возможен их вызов при работе программы. Следует иметь в виду, что протокол составлен таким образом, что любой из указанных вызовов может состояться, а может не произойти. При этом вызов вложенных подпрограмм невозможен, если не вызвана вышестоящая подпрограмма.

Пример

Входные данные	Выходные данные
4 PROC A CALL A ENDPROC PROC B CALL A ENDPROC PROC MAIN CALL B ENDPROC .	4 B B A B A A B A A A
3 PROC A CALL A ENDPROC PROC B CALL A ENDPROC PROC C ENDPROC PROC MAIN CALL A CALL B CALL C ENDPROC .	15 A A A A A A A A B A A C A B A B A A B C A C B B A B A A B A C B C C

Проверочные тесты

Входные данные	Ожидаемый результат
4 PROC A CALL A ENDPROC PROC B CALL A ENDPROC PROC MAIN CALL B ENDPROC .	4 B B A B A A B A A A

3 PROC A CALL A ENDPROC PROC B CALL A ENDPROC PROC C ENDPROC PROC MAIN CALL A CALL B CALL C ENDPROC .	15 A A A A A A A A B A A C A B A B A A B C A C B B A B A A B A C B C C
1 PROC A CALL A ENDPROC PROC MAIN CALL A ENDPROC .	1 A
10 PROC MAIN ENDPROC .	0
5 PROC A CALL B ENDPROC PROC B CALL A ENDPROC PROC MAIN CALL A CALL B ENDPROC .	14 A A B A B A A B A B A B A B A A B A B B A B B A B B A A B B A B B B A B A B B A B A B A B A B

Пример решения

```
n=int(input())
a=[]
while True:
```

```
    s=list(input().split())
    if s==['.']:
        break
    a.append(s)
#print(a)
start=a.index(['PROC','MAIN'])
ans=[]
import sys
sys.setrecursionlimit(10**9)
def plan(start,k,now,name):
    #print((start,k,now,name))
    global a,ans,n
    if 0<k<=n:
        now+=' '+name
        if now not in ans:
            ans.append(now)
    if k>n:
        return
    over=a[start:].index(['ENDPROC'])+start
    #print(over)
    do=[len(ans)]*(over-start)
    Do=len(ans)
    for i in range (start+1,over):
        yu=len(ans)
        to=a.index(['PROC',a[i][1]])
        for j in range (yu-1,Do-1,-1):
            bomb=ans[j].count(' ')
            if bomb<n:
                #for mn in range (i,over):
                    #if do[mn-start-1]>j:
                    #    continue
                plan(to,bomb+1,ans[j],a[to][1])
                    #yu=len(ans)
                    #do[mn-start-1]=yu
        plan(to,k+1,now,a[to][1])

    return
plan(start,0,'','MAIN')
print(len(ans))
#print(ans)
ans=sorted(ans)
for i in ans:
    print(i[1:])
```

Задача 6 (24 балла)

Условие

В ходе работы над оптимизацией компилятора для процессора из предыдущей задачи в фирме Хардсофт задумались об автоматическом выборе подпрограмм, которые станут инлайн-подпрограммами. Инлайн-подпрограммы - это такой способ сборки машинного кода, при котором тело подпрограммы во время компиляции подставляется непосредственно в место её вызова, и оно становится частью вызывающего кода. Таким образом, исключаются накладные расходы на вызов и завершение подпрограмм во время работы.

Инженеры фирмы Хардсофт считают, что инлайн-подпрограммами надо делать те, которые вызываются чаще всего. Однако вставка кода больших подпрограмм в места их вызова приведёт к резкому увеличению памяти, потребляемой программами, а для фирмы важна возможность лёгкой адаптации своего решения к промышленным устройствам, в которых значительную роль играет экономическая составляющая: в частности, необходимый для работы объём памяти должен быть минимальным.

Разработчики создали специальный анализатор, который определяет среднее количество запуска подпрограмм и размер каждой из них.

Требуется определить, какие из подпрограмм можно встроить в инлайн-режиме, для увеличения быстродействия и в пределах доступной памяти.

Входные данные: в первой строке через пробел указаны натуральные числа N, M и X. N и M не превышают 100 и являются количеством подпрограмм и количеством цепочек вызова, а X не превышает 10^6 и является количеством памяти (в килобайтах), доступным для размещения инлайн-подпрограмм. Далее идёт N строк, где через пробел перечислены имена подпрограмм (из латинских слов длиной до 20 символов) и их размер в килобайтах (натуральное число от 1 до 1000). Далее - M строк с описанием выявленных цепочек вызова: в них записаны имена подпрограмм через пробел, после которых также через пробел указано ожидаемое количество срабатываний этой цепочки за время работы программы (целое число от 1 до 1000).

Выходные данные: в первой строке - количество подпрограмм, которые можно пометить инлайновыми, в последующих строках - имена этих подпрограмм в том порядке, в котором они были указаны во входных данных.

Требуется выбрать подпрограммы так, чтобы количество вызовов инлайн-подпрограмм за время работы программы было максимальным.

Пример

Входные данные	Выходные данные
3 3 100 A 50 B 70 C 40 A 1 A B 4 B C 6	2 A C
3 3 100 A 50 B 70 C 40 A 1 A C 3 B 10	1 B

Проверочные тесты

Входные данные	Ожидаемый результат

3 3 100 A 50 B 70 C 40 A 1 A B 4 B C 6	2 A C
3 3 100 A 50 B 70 C 40 A 1 A C 3 B 10	1 B
1 1 10 A 20 A 1	0
3 5 50 A 8 B 16 C 32 A 1 B C 10 B C B 5 C B C 8 A C 2	2 B C
5 6 80 A 8 B 16 C 32 D 24 XYZ 48 XYZ B 1000 B C 10 B C B 5 C B C 8 A C 2 XYZ D 1000	3 A B XYZ

Пример решения

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <vector>
#include <map>
#include <set>
#include <iomanip>
```

```
#include <string>

using namespace std;

const int INF = 1e9;
const long double PI = acosl(-1);

map<string, int> rams;
map<vector<string>, int> used;
map<string, int> cnt;
vector<string> prognames;
vector<string> ans;
int maxcnt = 0, minweight = INF;
int n, m, ram;
vector< string > otv;

bool comp(string a, string b) {
    if (rams[a] == rams[b])
        return cnt[a] > cnt[b];
    return rams[a] < rams[b];
}

void rec(int totalram, int cntt, vector<string> ans, map<string, int>
used) {
    if (totalram > ram)
        return;
    if (cntt == maxcnt && minweight > totalram) {
        maxcnt = maxcnt;
        otv = ans;
        minweight = totalram;
    }
    else
        if (cntt > maxcnt) {
            maxcnt = cntt;
            otv = ans;
            minweight = totalram;
        }
    for (int i = 0; i < prognames.size(); i++) {
        auto to = prognames[i];
        if (used[to])
            continue;
        map<string, int> fused = used;
        vector<string> fans = ans;
        fans.push_back(to);
        fused[to]++;
        rec(totalram + rams[to], cntt + cnt[to], fans, fused);
    }
}

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
```

```
    cout.tie(0);
    cin >> n >> m >> ram;
    for (int i = 0; i < n; i++) {
        string s;
        int x;
        cin >> s >> x;
        rams[s] = x;
        prognames.push_back(s);
    }
    for (int i = 0; i < m; i++) {
        string s;
        getline(cin, s);
        if (s == "")
            getline(cin, s);
        vector<string> fnames;
        while (s.find(' ') != string::npos) {
            fnames.push_back(s.substr(0, s.find(' ')));
            s.erase(0, s.find(' ') + 1);
        }
        int num = stoi(s);
        for (auto i : fnames)
            cnt[i] += num;
    }
    for (int i = 0; i < prognames.size(); i++) {
        auto to = prognames[i];
        map<string, int> usd;
        usd[to] = 1;
        rec(rams[to], cnt[to], { to }, usd);
    }
    cout << otv.size() << '\n';
    for (auto i : otv) {
        cout << i << '\n';
    }
    return 0;
}
```