МУНИЦИПАЛЬНЫЙ ЭТАП ВСЕРОССИЙСКОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ В 2023-2024 УЧЕБНОМ ГОДУ ДЛЯ УЧАЩИХСЯ 9-11 КЛАССОВ

Время проведения - 240 минут (4 часа)
Максимальное количество баллов за задачу – 100
Максимальное количество баллов – 400

Особенности проведения - задания практического тура выполняются на компьютерах и сохраняются в отведенные папки. Не допускается использование локальной сети. Для проверки программ используется автоматическая тестирующая система Яндекс.Контест. Интернет-фильтр должен быть настроен только на использование соответствующих адресов и портов. Для авторизации должны быть подготовлены пароли и логины.

Проверяющая система:

https://official.contest.yandex.ru/contest/XXXX/enter
(здесь XXXX - номер контеста).

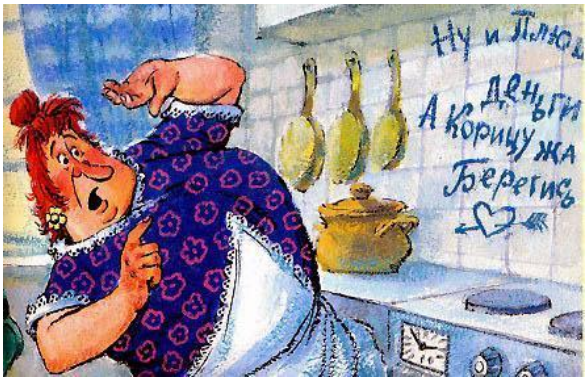
Ограничение по времени – 1 секунда
Ограничение по памяти – 64Mb
Ввод данных – через стандартный поток ввода или из файла input.txt
Вывод данных – через стандартный поток вывода или в файл output.txt

Критерии оценивания: *за каждый пройденный тест к задаче начисляется балл, который вычисляется по формуле: 100/"количество тестов"*

№	*1*	*2*	*3*	*4*
количество тестов	*10*	*10*	*10*	*10*
балл за тест	*10*	*10*	*10*	*10*

1. Плюшки

«Деньги дерешь, а корицу жалеешь! Берегись!», - именно эту фразу прочитала испуганная Фрекен Бок на стене своей кухни. И отправилась тренироваться в нелегком деле изготовления сладких булочек.



Карлсон считает, что корицы недостаточно, если масса корицы занимает не более $a\%$ от массы конкретной плюшки. И если хотя бы одна плюшка содержит слишком мало корицы, Карлсон расстроится. Помогите Фрекен Бок понять, будет ли доволен Карлсон в следующий раз.

Входные данные

Первая строка входного файла содержит количество плюшек n (целое число от 1 до 1000 включительно), и целое число a (в диапазоне от 1 до 100 включительно) – в каждой плюшке должно быть более $a\%$ корицы. На каждой из следующих n строк записаны через пробел два целых числа (целые числа в диапазоне от 1 до 1000) – масса очередной плюшки (включая корицу) и масса добавленной в нее корицы.

Выходные данные

Вывести слово «good» (строчными буквами без кавычек), если Карлсон будет доволен, слово «bad» (также строчными буквами без кавычек) в противном случае. Карлсон будет доволен, если каждая плюшка содержит корицы более $a\%$ от массы всей плюшки.

Примеры

Входные данные	**Выходные данные**
2 5 100 10 200 10	bad
2 1 50 5 100 2	good

Пояснение

В первом примере Карлсона расстроит уровень корицы во второй плюшке – 5%, тогда как ему необходимо более 5%. Во втором примере содержание корицы 10% в первой плюшке и 2% во второй, что больше необходимого 1%.

Решение:

```
n, a = map(int, input().split())
plu = [0]*n
cor = [0]*n
for i in range(n):
    plu[i], cor[i] = map(int, input().split())

good = True
for i in range(n):
    if cor[i]*100 <= plu[i]*a:
        good = False

if good:
    print("good")
else:
    print("bad")
```

2. Банки с вареньем

Малыш в ожидании Карлсона хочет украсить симпатичными надписями банки с любимым вареньем. Для этого Малыш собирается написать на каждой банке название варенья красивым каллиграфическим почерком. Посчитайте, сколько времени займут приготовления Малыша.

Входные данные

На первой строке входного файла задано целое число $1 \le n \le 26$ – количество букв, которые Малыш умеет красиво рисовать. На каждой из следующих n строк указана сама буква (строчная латинская) и через пробел время в минутах (целое число в диапазоне от 1 до 100) на то, чтобы ее красиво нарисовать.

На следующей строке располагается одно целое число $1 \le m \le 20$ – количество различных видов варенья. А на каждой из следующих m строк написано сначала название очередного варенья (слово состоит строчных латинских букв, длина слова не больше 10), а затем через пробел количество банок с этим вареньем (целое число в диапазоне от 1 до 100).

Гарантируется, что все названия содержат только те буквы, которые умеет рисовать Малыш.

Выходные данные

На первой строке выходного файла необходимо вывести одно целое число – количество минут для того, чтобы на каждой банке Малыш написал название варенья. Время тратится только на написание букв.

На следующих строках, по одной на каждый тип варенья, необходимо вывести название варенья и через пробел общее количество минут, необходимое, чтобы на каждой банке данного типа нарисовать надпись. Названия типов варенья необходимо вывести в алфавитном порядке

Пример

Входные данные	Выходные данные
9 m 2 a 3 l 1 i 1 n 2 s 1 o 5 r 1 d 1 2 smorodina 1 malina 2	45 malina 24 smorodina 21

Решение:

```
n = int(input())
letters = {}
for i in range(n):
    key, time = input().split()
    letters[key] = int(time)

m = int(input())
names = {}
for i in range(m):
    name, count = input().split()
    names[name] = int(count)

times = {}
for name in names:
    s = 0
    for letter in name:
        s += letters[letter]
    times[name] = s*names[name]

snames = list(times.keys())
snames.sort()
sorted_times = {name: times[name] for name in snames}

print(sum(times.values()))
for name, time in sorted_times.items():
    print(name, time)
```

3. Тефтели

Карлсон улетел… Но он обещал вернуться! В связи с чем Мама Малыша каждую неделю готовит тефтели, а Малыш потом красиво раскладывает их в ряд по тарелочкам – вдруг Карлсон неожиданно вернется, а угощение уже готово! Тарелочки бывают большими и маленькими, и Мама уже расставила их на столе в один ряд. На каждой большой тарелочке можно разместить не более k тефтелей, на каждой маленькой – не более m тефтелей.

Сколькими способами Малыш сможет это сделать, если на каждой тарелочке должна лежать хотя бы одна тефтеля?

Входные данные

На первой строке входного файла расположены через пробел три целых числа $1 \le n \le 100$, $1 \le k \le 20$, $1 \le m \le 20$ – соответственно общее число тефтелей, максимальное количество тефтелей, помещающихся на большой тарелочке, и максимальное количество тефтелей на маленькой тарелочке, $m \le k$.

На второй строке расположены через пробел несколько чисел 1 и/или 2 в некотором порядке. Количество этих чисел совпадает с числом тарелочек на столе. Число «1» означает, что соответствующая тарелочка является большой, а число «2» – маленькой.

Гарантируется, что тарелочек достаточно, чтобы разместить все тефтели с указанными ограничениями, кроме того общее число тарелочек не превосходит 16, и на столе присутствует хотя бы одна тарелочка.

Выходные данные

Необходимо вывести одно целое число – количество способов разместить все тефтели по тарелочкам.

Пример

Входные данные	Выходные данные
2 5 3 1 2	1
11 5 3 1 2 2	1
5 3 2 1 2 2	4

Пояснение

В первом примере всего две тефтели и две тарелочки. Есть только один способ расположить тефтели так, чтобы на каждой тарелочке была хотя бы одна тефтеля: 1 – 1, то есть одна на первой тарелочке, и одна на второй.

Во втором примере тефтелей 11, а тарелочки расположены следующим образом – большая первая, а вторая и третья маленькие. Учитывая, что вместимость большой 5, а вместимость маленькой 3, получаем что 11 – максимальное количество тефтелей, которые можно разместить на этих трех тарелочках. Тогда способ размещения всех тефтелей только один: 5 – 3 – 3, то есть на первой 5, а на второй и третьей – по три.

Способы размещения в третьем примере: 1 – 2 – 2, 2 – 1 – 2, 2 – 2 – 1, 3 – 1 – 1.

Решение:

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
using namespace std;

int main()
{
    string line;
    istringstream stream;

    int n, k, m, a;
    getline(cin, line);
    stream.str(line);
    stream >> n >> k >> m;

    vector<int> plates;
    getline(cin, line);
```

```
    stream.clear();
    stream.str(line);
    while (stream >> a) {
        plates.push_back( a==1?k:m );
    }
    int p = plates.size();

    vector< vector<long long> > tef(p);
    for(int i=0; i<p; ++i) {
        tef[i].resize(n+1);
        tef[i][0] = 0;
    }

    for(int i=1; (i<=plates[0]) && (i<=n); ++i) {
        tef[0][i] = 1;
    }

    for(int platesNumber=1; platesNumber<p; ++platesNumber) {
        for(int tefCount=1; tefCount<=n; ++tefCount) {
            long long ways = 0;
            for(int tefCountOnPlate=1;
(tefCountOnPlate<=plates[platesNumber])&&(tefCount>=tefCountOnPlate);
++tefCountOnPlate) {
           ways += tef[platesNumber-1][tefCount - tefCountOnPlate];
            }
            tef[platesNumber][tefCount] = ways;
        }
    }
    cout << tef[p-1][n] << endl;
    return 0;
}
```

4. Чемпионат по поеданию пирогов

Одним ясным летним вечером, Карлсон решил поучаствовать в чемпионате по поеданию пирогов, варенья и прочих вкусностей. Но так, чтобы его команда непременно выиграла! Идея соревнования заключается в том, что каждый участник на своем участке занимается поеданием самого своего любимого лакомства. На каждом участке свой вид лакомства. Участники действуют по очереди. Как только закончил один – начинает другой. Всего в соревновании N участникам необходимо съесть N видов вкусностей на N участках (по одному виду на каждом участке). Однако каждый участник может съесть лишь ограниченное количество пирогов, варенья и прочих вкусностей. В ходе долгих тренировок способности каждого участника команды Карлсона к поеданию каждой вкусности стали известны. Помоги Карлсону распределить каждого участника на свой участок поедания так, чтобы эффективность команды была максимальной (чтобы они съели как можно большее суммарное количество вкусностей).

Входные данные

На первой строке входного файла записано число N – количество вкусностей для поедания (участников соревнования столько же), $1 \le N \le 40$. Далее задана матрица A[N , N], где Aij – способность i-ого участника съесть j-ю вкусность.

Выходные данные

Необходимо последовательно вывести через пробел номера участков для каждого участника.

Пример

Входные данные	Выходные данные
3 10 10 15 20 5 10 5 7 8	3 1 2

Пояснение к примеру:

Первого участника лучше всего отправить есть вкусность номер 3, так как здесь он сможет съесть 15 вкусностей типа 3. Второй участник сможет съесть 20 вкусностей типа 1. Третьего участника отправят есть вкусности номер 2 – их он съест 7. Максимальное суммарное количество вкусностей: 15+20+7=42.

Решение:

```
#include <iostream>
#include <vector>
using namespace std;
int N;
vector< vector<int> > a;
const int INF = 10000;
int main()
{
    cin >> N;
    a.resize(N+1);
    for(int i=0; i<N+1; ++i)
    {
        a[i].resize(N+1);
    }
    for(int i=1; i<=N; ++i)
    {
        for(int j=1; j<=N; ++j)
        {
            cin >> a[i][j];
           a[i][j] = -a[i][j];
        }
    }
    vector<int> u (N+1), v (N+1), p (N+1), way (N+1);
    for (int i=1; i<=N; ++i)
    {
```

```
        p[0] = i;
        int j0 = 0;
        vector<int> minv (N+1, INF);
        vector<char> used (N+1, false);
        do {
                used[j0] = true;
                int i0 = p[j0],  delta = INF,  j1;
                for (int j=1; j<=N; ++j)
                        if (!used[j])
                        {
                                int cur = a[i0][j]-u[i0]-v[j];
                                if (cur < minv[j])
                                        minv[j] = cur,  way[j] = j0;
                                if (minv[j] < delta)
                                        delta = minv[j],  j1 = j;
                        }
                for (int j=0; j<=N; ++j)
                        if (used[j])
                                u[p[j]] += delta,  v[j] -= delta;
                        else
                                minv[j] -= delta;
                j0 = j1;
        } while (p[j0] != 0);
        do
        {
                int j1 = way[j0];
                p[j0] = p[j1];
                j0 = j1;
        } while (j0);
    }
    vector<int> ans (N+1);
    for (int j=1; j<=N; ++j)
            ans[p[j]] = j;
        for(int i=1; i<=N; ++i)
        {
            cout << ans[i] << ' ';
        }
        cout << endl;
}
```