Олимпиада школьников «Шаг в будущее»
Заключительный этап

11 класс
Вариант 1

Задача 1 (8 баллов)

Условие

Для заданного натурального числа, записанного в десятичной системе счисления, требуется определить, является ли двоичная запись этого числа чередующейся последовательностью нулей и единиц.

Входные данные: число N, не превышающее 10^6.

Выходные данные: слово YES, если двоичная запись числа - чередующаяся последовательность нулей и единиц, или NO в противном случае.

Пример

Входные данные	Выходные данные
5	YES
3	NO
10	YES

Проверочные тесты

Входные данные	Ожидаемый результат
5	YES
3	NO
10	YES
1000000	NO
2	YES
699050	YES
349525	YES
524287	NO
524288	NO

Пример решения

```
def sol(n):
    last = None
```

```
        while n > 0:
        if last == n % 2:
        return "NO"

        last = n % 2
        n //= 2

        return "YES"

print(sol(int(input())))
```

Задача 2 (12 баллов)

Условие

При взаимодействии по сети большинство программ обмениваются информацией в виде пакетов – блоков данных определённой длины. При этом на более низком уровне сетевых протоколов пакеты могут делиться на части или объединяться, в зависимости от аппаратных возможностей и прочих факторов. В конечном итоге к программе-получателю пакеты могут приходить тоже частями, либо по несколько сразу.

Требуется написать программу, которая для заданной последовательности байтов, поступающей к получателю и описывающей набор пакетов с массивами целых положительных чисел, выявит пакет с самой большой суммой элементов.

Входные данные: в каждой строке записано по целому числу от 0 до 255, количество строк не превышает 10000. Числа соответствуют байтам пакетов, поступивших к получателю, по порядку их отправки. Каждый i-й пакет начинается с двух байтов, хранящих его длину N_i в виде 16-битного числа - сначала старший байт, затем младший. Затем идёт N_i байтов - чисел, включённых в пакет. Ввод завершается строкой, состоящей из одной точки.

Выходные данные: через пробел в строку должны быть выведены 2 числа - номер пакета, содержащего массив чисел с самой большой суммой элементов (нумерацию вести с 1), и сумма чисел в этом массиве.

Пример

Входные данные	Выходные данные
0 2 1 2 0 3 200 210 190 0 1 100 .	2 600
0 1 10 0 1 9 .	1 10

Проверочные тесты

Входные данные	Ожидаемый результат
0 2 1 2 0 3 200 210 190 0 1 100 .	2 600

0 1 10 0 1 9 .	1 10
0 1 123 .	1 123
1 0 (256 строк с цифрой 1) 0 2 123 123 .	1 256
1 0 (256 строк с цифрой 1) 0 2 123 123 .	2 258
(длинный тест)	10 400

Пример решения

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

using namespace std;

void solve() {
	int a, b;
	int ans = 0;
```

```
        int ans_k = 0;
        int k = 0;
        while (cin >> a >> b) {
        k += 1;
        int n = a * 256 + b;
        int s = 0;
        for (int i = 0; i < n; ++i) {
        int tmp;
        cin >> tmp;
        s += tmp;
        if (s > ans) {
                ans = s;
                ans_k = k;
        }
        }
        }
        cout << ans_k << ' ' << ans << '\n';
}

signed main() {
        solve();
}
```

Задача 3 (16 баллов)

Условие

Разреженной матрицей называется такая матрица, в которой большинство элементов равны нулю. Для экономии памяти при хранении разреженных матриц большого объема используются специальные структуры данных, например, тройки из индекса строки, индекса столбца и значения для каждого ненулевого элемента.

Подматрицей в рамках данной задачи называется такая матрица, которая является непрерывным фрагментом исходной матрицы и может быть получена исключением ряда крайних строк и столбцов.

Требуется написать программу для определения наибольшей по количеству элементов нулевой подматрицы в разреженной матрице.

Входные данные: в первой строке через пробел записаны количество строк разреженной матрицы N (<= 10^6), количество столбцов M (<=10^6) и количество ненулевых элементов K (<=10^3).

Далее в K строках через пробел записаны по 3 числа: номер строки, номер столбца (нумерация идёт с 1) и целое значение элемента (модуль не превышает 1000).

Выходные данные: в первой строке - индексы строки и столбца левого верхнего угла искомой подматрицы, записанные через пробел, во второй строке - индексы строки и столбца правого нижнего угла искомой подматрицы, записанные через пробел. Если

существует несколько вариантов с одинаковым количеством элементов - вывести любой.

Пример

Входные данные	Выходные данные
100 100 2 1 1 5 100 90 5	1 2 99 100
1000 1000 3 1 1 1 1 2 2 1 3 3	2 1 1000 1000

Проверочные тесты

Входные данные	Ожидаемый результат
100 100 2 1 1 5 100 90 5	1 2 99 100
1000 1000 3 1 1 1 1 2 2 1 3 3	2 1 1000 1000
10 10 0	1 1 10 10
2 2 3 1 1 1 1 2 1 2 1 1	2 2 2 2
3 3 2 3 2 10 3 3 100	1 1 2 3

1000000 1000000 5 1 1 1 2 2 1 3 3 1 4 4 1 5 100000 1	6 1 1000000 1000000
1000000 1000000 5 1 1 1 2 2 1 3 3 1 4 4 1 1000000 100000 1	1 5 999999 1000000
1000000 1000000 7 1 1 1 2 2 1 3 3 1 4 4 1 1000 1000 1 2000 3000 11 1000000 100000 1	2001 1 999999 1000000

Пример решения

```
n, m, count=map(int, input().split())
a=[]
for i in range(count):
        a.append(list(map(int, input().split())))
max1=[0, 0]
max2=[0, 0]
maxsq=0
checkx=[1, n]
checky=[1, m]
for i in range(count):
        checkx.append(a[i][0])
        if a[i][0]<n:
        checkx.append(a[i][0]+1)
        if a[i][0]>1:
        checkx.append(a[i][0]-1)
        checky.append(a[i][1])
        if a[i][1] < m:
        checky.append(a[i][1] + 1)
        if a[i][1] > 1:
        checky.append(a[i][1] - 1)
```

```
checkx.sort()
checky.sort()
for x1 in checkx:
        for y1 in checky:
        for x2 in checkx:
        if x1>x2:
                continue
        for y2 in checky:
                if y1>y2:
                continue
                flag=0
                for i in range(count):
                if a[i][0]>=x1 and a[i][0]<=x2 and a[i][1]>=y1 and a[i][1]<=y2:
                flag=1
                break
                if flag==1:
                continue
                sq=(x2-x1+1)*(y2-y1+1)
                if sq>maxsq:
                maxsq=sq
                max1=[x1, y1]
                max2=[x2, y2]
print(*max1)
print(*max2)
```

Задача 4 (20 баллов)

Условие

Большинство баз данных строится на основе принципа представления информации в табличном виде, при этом различные таблицы используются для представления сущностей разных типов. Столбцами таблиц являются атрибуты сущностей (также они называются **полями**), а строками - информация о самих сущностях (они называются **записями**).

Таблицы нередко являются связанными, что отражает взаимодействие представляемых ими сущностей.

Рассмотрим в качестве примера базы данных список учеников города Энска для электронного дневника. Такая база данных может быть представлена тремя сущностями: школа, класс, ученик.

Сущность "Школа" описывается двумя полями:

- идентификатор школы (натуральное число от 1 до 10^5);
- название (строка из латинских букв и цифр длиной от 1 до 20 знаков).

Сущность "Класс" описывается тремя полями:

- идентификатор класса (натуральное число от 1 до 10^5);

- идентификатор школы;
- название (строка из латинских букв и цифр длиной от 1 до 5 знаков).

Сущность "Ученик" описывается 4 полями:

- идентификатор ученика (натуральное число от 1 до 10^5);
- идентификатор класса;
- фамилия (строка из латинских букв и цифр длиной от 1 до 20 знаков);
- имя (строка из латинских букв и цифр длиной от 1 до 20 знаков).

Идентификаторы в пределах одного типа сущностей не повторяются, а идентификаторы, являющиеся ссылками на другие сущности, гарантированно существуют в соответствующих таблицах.

Часть школ в базе данных может не содержать ни одного класса (например, находится на ремонте), а часть классов может не содержать ни одного ученика (например, класс только формируется или все ученики из него выпустились).

Требуется сформировать соединение всех сведений об учениках (школа - класс - ученик), но так, чтобы в него попали сущности без дочерних элементов (школы без классов и классы без учеников) тоже.

Входные данные: в первой строке записано натуральное число S - количество школ (не превышает 10^3). Далее в S строках через пробел записаны идентификатор и название каждой школы. Затем в следующей строке записано натуральное число C - количество классов (не превышает 10^3). Далее в C строках через пробел для каждого класса записаны его идентификатор, идентификатор школы и название. Затем в следующей строке записано натуральное число P - количество учеников (не превышает 10^4). Далее в P строках через пробел для каждого ученика записаны его идентификатор, идентификатор класса, фамилия и имя.

Выходные данные: искомое соединение, где по строкам записана информация об учениках через пробел:

- идентификатор школы;
- название школы;
- идентификатор класса;
- название класса;
- идентификатор ученика;
- фамилия ученика;
- имя ученика.

Если в школе нет ни одного класса - итоговая информация о ней должна состоять из одной строки, где через пробел будут указаны только идентификатор и название школы, хвостовых пробелов быть не должно.

Если в классе нет ни одного ученика - итоговая информация о нём должна состоять из одной строки, где через пробел будут указаны только идентификатор и название школы, идентификатор и название класса, хвостовых пробелов быть не должно.

Результирующие данные должны быть отсортированы по возрастанию по идентификатору школы, затем по идентификатору класса, затем - по идентификатору ученика.

Пример

Входные данные	Выходные данные
2 1 School1 2 School2 4 1 1 11a 2 1 11b 3 2 11a 4 2 11b 7 1 1 Ivanov Ivan 2 1 Petrov Petr 3 2 Aleshin Aleksey 4 2 Vasiliev Vasiliy 5 3 Semenov Semen 6 3 Ivanov Petr 7 3 Petrov Semen	1 School1 1 11a 1 Ivanov Ivan 1 School1 1 11a 2 Petrov Petr 1 School1 2 11b 3 Aleshin Aleksey 1 School1 2 11b 4 Vasiliev Vasiliy 2 School2 3 11a 5 Semenov Semen 2 School2 3 11a 6 Ivanov Petr 2 School2 3 11a 7 Petrov Semen 2 School2 4 11b
2 1 School1 2 School2 2 2 2 11b 1 2 11a 4 4 2 Vasiliev Vasiliy 1 1 Ivanov Ivan 3 2 Aleshin Aleksey 2 1 Petrov Petr	1 School1 2 School2 1 11a 1 Ivanov Ivan 2 School2 1 11a 2 Petrov Petr 2 School2 2 11b 3 Aleshin Aleksey 2 School2 2 11b 4 Vasiliev Vasiliy

Олимпиада школьников «Шаг в будущее»
Заключительный этап

Проверочные тесты

Входные данные	Ожидаемый результат
2 1 School1 2 School2 4 1 1 11a 2 1 11b 3 2 11a 4 2 11b 7 1 1 Ivanov Ivan 2 1 Petrov Petr 3 2 Aleshin Aleksey 4 2 Vasiliev Vasiliy 5 3 Semenov Semen 6 3 Ivanov Petr 7 3 Petrov Semen	1 School1 1 11a 1 Ivanov Ivan 1 School1 1 11a 2 Petrov Petr 1 School1 2 11b 3 Aleshin Aleksey 1 School1 2 11b 4 Vasiliev Vasiliy 2 School2 3 11a 5 Semenov Semen 2 School2 3 11a 6 Ivanov Petr 2 School2 3 11a 7 Petrov Semen 2 School2 4 11b
2 1 School1 2 School2 2 2 2 11b 1 2 11a 4 4 2 Vasiliev Vasiliy 1 1 Ivanov Ivan 3 2 Aleshin Aleksey 2 1 Petrov Petr	1 School1 2 School2 1 11a 1 Ivanov Ivan 2 School2 1 11a 2 Petrov Petr 2 School2 2 11b 3 Aleshin Aleksey 2 School2 2 11b 4 Vasiliev Vasiliy
1 1 School1 1 2 1 11a 1 3 2 Ivanov Ivan	1 School1 2 11a 3 Ivanov Ivan

Олимпиада школьников «Шаг в будущее»
Заключительный этап

2 1 School1 10 School2 2 2 1 11a 20 1 1 1 3 2 Ivanov Ivan	1 School1 2 11a 3 Ivanov Ivan 1 School1 20 1 10 School2
3 11 School1 12 Abc 13 School2 2 100 12 Abc1 101 13 10z 3 5 101 Ivanov Ivan 6 101 Petrov Petr 7 101 Ivanov Ivan	11 School1 12 Abc 100 Abc1 13 School2 101 10z 5 Ivanov Ivan 13 School2 101 10z 6 Petrov Petr 13 School2 101 10z 7 Ivanov Ivan
3 13 School2 12 Abc 11 School1 2 101 13 10z 100 12 Abc1 3 7 101 Ivanov Ivan 6 101 Petrov Petr 5 101 Ivanov Ivan	11 School1 12 Abc 100 Abc1 13 School2 101 10z 5 Ivanov Ivan 13 School2 101 10z 6 Petrov Petr 13 School2 101 10z 7 Ivanov Ivan

2 1 School 2 School 4 1 1 11 2 1 11 3 2 11 4 2 11 7 1 1 Ivanov Ivan 2 1 Petrov Petr 3 2 Aleshin Aleksey 4 2 Vasiliev Vasiliy 5 3 Semenov Semen 6 3 Ivanov Petr 7 3 Petrov Semen	1 School 1 11 1 Ivanov Ivan 1 School 1 11 2 Petrov Petr 1 School 2 11 3 Aleshin Aleksey 1 School 2 11 4 Vasiliev Vasiliy 2 School 3 11 5 Semenov Semen 2 School 3 11 6 Ivanov Petr 2 School 3 11 7 Petrov Semen 2 School 4 11
5 1 School 3 A 4 B 5 C 2 School 6 1 1 11 2 1 11 3 2 11 4 2 11 5 5 10 6 4 9 8 1 1 Ivanov Ivan 2 1 Petrov Petr 3 2 Aleshin Aleksey 4 2 Vasiliev Vasiliy 5 3 Semenov Semen 6 3 Ivanov Petr 7 3 Petrov Semen 8 6 Sidorova Sonya	1 School 1 11 1 Ivanov Ivan 1 School 1 11 2 Petrov Petr 1 School 2 11 3 Aleshin Aleksey 1 School 2 11 4 Vasiliev Vasiliy 2 School 3 11 5 Semenov Semen 2 School 3 11 6 Ivanov Petr 2 School 3 11 7 Petrov Semen 2 School 4 11 3 A 4 B 6 9 8 Sidorova Sonya 5 C 5 10

Пример решения

```
d = dict()
c = dict()
```

```
school = dict()
student = dict()
for q in range(3):
        n = int(input())
        for j in range(n):
        s = input().split()
        if q == 0:
        d[int(s[0])] = []
        school[int(s[0])] = s[1]
        elif q == 1:
        d[int(s[1])].append([int(s[0]), []])
        c[int(s[0])] = [int(s[1]), s[2]]
        else:
        student[int(s[0])] = [int(s[1]), s[2] + ' ' + s[3]]
        for x in d[c[int(s[1])][0]]:
                if x[0] == int(s[1]):
                x[1].append(int(s[0]))

#print(d)

res = []

for k in d.keys():
        a = [k, 0, 0]
        if len(d[k]) == 0:
        b = a.copy()
        res.append(b)
        for x in d[k]:
        a[1] = x[0]
        if len(x[1]) == 0:
        b = a.copy()
        res.append(b)
        for s in x[1]:
        a[2] = s
        b = a.copy()
        res.append(b)
        a[1] = 0
        a[2] = 0



res.sort()

# print(d)
```

```
# print(res)

for x in res:
        if x[0] != 0 and x[1] != 0 and x[2] != 0:
        print(x[0], school[x[0]], x[1], c[x[1]][1], x[2], student[x[2]][1])
        elif x[0] != 0 and x[1] != 0:
        print(x[0], school[x[0]], x[1], c[x[1]][1])
        else:
        print(x[0], school[x[0]])
```

Задача 5 (20 баллов)

Условие

Система управления базами данных (СУБД) - программа, обеспечивающая управление созданием и использованием баз данных.

Большинство СУБД предназначено для одновременной работы с ними нескольких пользователей. Это значит, что запросы на доступ к сущностям могут поступать сразу от нескольких клиентов СУБД. Для исключения ситуаций некорректного одновременного редактирования данных разными пользователями разработан механизм блокировок. Он позволяет клиенту СУБД "заблокировать" (установить блокировку) некоторой сущности, чтобы другие не имели к ней доступа на время обработки. В случае, если к заблокированной сущности обращается другой клиент, его работа приостанавливается до тех пор, пока блокировка не будет снята. (Понятно, что блокировки должны устанавливаться на как можно более короткое время, чтобы не замедлять работу других пользователей.)

В дальнейшем будем называть программы, обслуживающие запросы клиентов, **процессами**, а сущности - **ресурсами**.

Одна из проблем, связанных с блокировками - возникновение взаимных блокировок (deadlock). Они могут возникать, когда процессы запрашивают сразу несколько блокировок, но делают это в разном порядке. Например, процесс 1 заблокировал ресурс A, а процесс 2 заблокировал ресурс B. После этого процесс 1 отправляет запрос на блокировку ресурса B и попадает в состояние ожидания, пока процесс 2 освободит эту ресурс. Если вместо этого процесс 2 запросит блокировку ресурса A - он тоже попадёт в состояние ожидания, которое никогда не закончится.

Требуется написать программу, которая по журналу действий процессов определит, возникла ли взаимоблокировка в процессе работы, или нет.

Входные данные: в строках через пробел записаны номер процесса (натуральное число, не превышающее 100), заглавная латинская буква, обозначающая ресурс, и буква, обозначающая действие: L - запрос на блокировку ресурса, U - снятие блокировки. Ввод завершается строкой, состоящей из одной точки.

Журнал не содержит противоречий, в частности, процессы не могут освобождать ресурсы, которые они не заняли, а также процессы не совершают действий, если находятся в ожидании освобождения блокировок.

Процесс не может установить на ресурс новую блокировку, не сняв предыдущую.
Выходные данные: если в процессе работы случилась взаимоблокировка - вывести в первой строке слово DEADLOCK, а во второй строке через пробел - номера процессов по возрастанию, участвующих в этой взаимоблокировке. Если в журнале обнаруживается несколько взаимоблокировок, вывести первую из них. В случае, если взаимоблокировок не случалось, вывести в первой строке текст "NO DEADLOCK" без кавычек, а во второй строке - количество ресурсов, которые остались заблокированными к концу журнала.

Пример

Входные данные	Выходные данные
1 A L 2 B L 1 A U 2 B U .	NO DEADLOCK 0
1 A L 2 B L 3 Z L 2 Z L 3 A L 1 Z L .	DEADLOCK 1 3

Проверочные тесты

Входные данные	Ожидаемый результат
1 A L 2 B L 1 A U 2 B U .	NO DEADLOCK 0
1 A L 2 B L 3 Z L 2 Z L 3 A L 1 Z L .	DEADLOCK 1 3

10 A L 10 A U .	NO DEADLOCK 0
10 A L 10 A U 1 A L 1 A U 10 B L 10 B U 2 B L 2 B U .	NO DEADLOCK 0
10 Z L 9 Y L 9 Z L 10 Y L .	DEADLOCK 9 10
99 B L 98 A L 100 L L 98 A U .	NO DEADLOCK 2
1 A L .	NO DEADLOCK 1
1 A L 2 B L 3 C L 2 C L 3 B L .	DEADLOCK 2 3
1 A L 2 B L 3 C L 2 B U 3 B L .	NO DEADLOCK 3

1 A L 2 B L 3 C L 3 A L 1 B L 2 C L .	DEADLOCK 1 2 3

Пример решения

```
#include <iostream>
#include <string>
#include <algorithm>
#include <set>
#include <map>
using namespace std;

struct process_t{
   set<char> locked;
};

struct resource_t{
   set<int> locked_by;
};

process_t processes[100];
map<char,resource_t> resources;



set<int> deadlocked;

bool checkProcesses(set<int> checkedPids, int prevRid, int pid){
   for(int checkedPid : checkedPids){
         if(pid == checkedPid) {
               checkedPids.emplace(pid);
               deadlocked = checkedPids;
               return true;
         }
   }
   checkedPids.emplace(pid);
   for(int rid : processes[pid].locked){
         if(rid == prevRid) continue;
         for(int nextPid : resources[rid].locked_by){
```

```
                if(nextPid == pid) continue;
                if(checkProcesses(checkedPids, rid, nextPid)){
                        return true;
                }
        }
    }
    return false;
}

bool checkDeadlocks(){
    for(int pid = 0;pid<100;pid++){
        for(int rid : processes[pid].locked){
                for(int nextPid : resources[rid].locked_by){
                        if(nextPid == pid) continue;
                        set<int> checkedPid;
                        checkedPid.emplace(pid);
                        if(checkProcesses(checkedPid,rid,nextPid)){
                                return true;
                        }
                }
        }
    }
    return false;
}

int main()
{
    while(true){
        char lineChars[15];
        cin.getline(lineChars,14);
        string line = string(lineChars);
        if(line.length() == 0)continue;
        if(line[0] == '.') break;
        int pid = stoi(line.substr(0,line.length()-4))-1;
        char rid = line[line.length()-3];
        char action = line[line.length()-1];

        if(action == 'U'){
                processes[pid].locked.erase(rid);
                resources[rid].locked_by.erase(pid);
        }
        if(action == 'L'){
                processes[pid].locked.emplace(rid);
                resources[rid].locked_by.emplace(pid);
```

```
        }
        if(checkDeadlocks()){
                cout << "DEADLOCK\n";
                for(int lockedPid : deadlocked){
                        cout << lockedPid+1 << " ";
                }
                cout << endl;
                while(true){
                        cin.getline(lineChars,14);
                        string line = string(lineChars);
                        if(line[0] == '.') break;
                }
                return 0;
        }
    }

    cout << "NO DEADLOCK\n";
    int lockedCount = 0;
    for(pair<char,resource_t> r : resources){
        if(r.second.locked_by.size()>0)lockedCount++;
    }
    cout << lockedCount << endl;
    return 0;
}
```

Задача 6 (24 баллов)

Условие

Требуется провести расширенный анализ взаимоблокировок, описанных в предыдущей задаче, с учётом следующей особенности: блокировки ресурсов могут иметь 2 типа - блокировка для чтения и блокировка для записи.

Установка процессом блокировки ресурса только для чтения означает, что процесс не планирует изменять значение ресурса, и не мешает другим процессам устанавливать такую же блокировку. Однако блокировка для записи невозможна, если на ресурс уже установлена блокировка любого типа другим процессом, и приводит к приостановке работы процесса, пытающегося такую блокировку установить. Также установленная блокировка для записи не позволит другим процессам взять блокировку этого ресурса для чтения и приведёт к их приостановке.

Входные данные: в строках через пробел записаны номер процесса (натуральное число, не превышающее 100), заглавная латинская буква, обозначающая ресурс, и буква, обозначающая действие: R - запрос на блокировку ресурса для чтения, W - запрос на

блокировку ресурса для записи, U - снятие блокировки. Ввод завершается строкой, состоящей из одной точки.

Журнал не содержит противоречий, в частности, процессы не могут освобождать ресурсы, которые они не заняли, а также процессы не совершают действий, если находятся в ожидании освобождения блокировок.

Процесс не может установить на ресурс новую блокировку, не сняв предыдущую.

Выходные данные: если в процессе работы случилась взаимоблокировка - вывести в первой строке слово DEADLOCK, а во второй строке через пробел - номера процессов по возрастанию, участвующих в этой взаимоблокировке. Если в журнале обнаруживается несколько взаимоблокировок, вывести первую из них. В случае, если взаимоблокировок не случалось, вывести в первой строке текст "NO DEADLOCK" без кавычек, а во второй строке - количество ресурсов, которые остались заблокированными к концу журнала.

Пример

Входные данные	Выходные данные
1 A W 2 B W 1 A U 2 B U .	NO DEADLOCK 0
1 A R 2 B W 3 Z R 2 Z W 3 A W 1 Z W .	DEADLOCK 1 3

Проверочные тесты

Входные данные	Ожидаемый результат
1 A W 2 B W 1 A U 2 B U .	NO DEADLOCK 0

1 A R 2 B W 3 Z R 2 Z W 3 A W 1 Z W .	DEADLOCK 1 3
10 A W 10 A U .	NO DEADLOCK 0
10 A W 10 A U 1 A W 1 A U 10 B W 10 B U 2 B W 2 B U .	NO DEADLOCK 0
10 Z W 9 Y W 9 Z W 10 Y W .	DEADLOCK 9 10
99 B W 98 A W 100 L W 98 A U .	NO DEADLOCK 2
1 R .	NO DEADLOCK 1
1 A W 2 B W 3 C W 2 C W 3 B R .	DEADLOCK 2 3

1 A W 2 B W 3 C W 2 B U 3 B R .	NO DEADLOCK 3
1 A W 2 B W 3 C W 3 A W 1 B W 2 C W .	DEADLOCK 1 2 3
10 Z W 9 Y R 9 Z R 10 Y W .	DEADLOCK 9 10
1 A R 2 B W 3 C R 3 A W 1 B R 2 C W .	DEADLOCK 1 2 3

Пример решения

```
#include <iostream>
#include <string>
#include <algorithm>
#include <set>
#include <map>
using namespace std;


struct process_t{
   set<pair<char,bool>> locked;
};

struct resource_t{
   set<pair<int,bool>> locked_by;
```

```
};

process_t processes[100];
map<char,resource_t> resources;



set<int> deadlocked;

bool checkProcesses(set<int> checkedPids, int prevRid, int pid, bool onlyReadLocks){
   for(int checkedPid : checkedPids){
         if(pid == checkedPid) {
               if(!onlyReadLocks){
               checkedPids.emplace(pid);
               deadlocked = checkedPids;
               return true;
               }else{
                     return false;
               }
         }
   }
   checkedPids.emplace(pid);
   for(pair<char,bool> rid : processes[pid].locked){
         if(rid.first == prevRid) continue;
         for(pair<int,bool> nextPid : resources[rid.first].locked_by){
               if(nextPid.first == pid) continue;
               if(checkProcesses(checkedPids, rid.first, nextPid.first, onlyReadLocks &&
rid.second)){
                     return true;
               }
         }
   }
   return false;
}

bool checkDeadlocks(){
   for(int pid = 0;pid<100;pid++){
         for(pair<char,bool> rid : processes[pid].locked){
               for(pair<int,bool> nextPid : resources[rid.first].locked_by){
                     if(nextPid.first == pid) continue;
                     set<int> checkedPid;
                     checkedPid.emplace(pid);
                     bool onlyReadLocks = rid.second;
```

```
                        if(checkProcesses(checkedPid,rid.first,nextPid.first,
onlyReadLocks)){
                                return true;
                        }
                }
        }
    }
    return false;
}

int main()
{
    while(true){
        char lineChars[15];
        cin.getline(lineChars,14);
        string line = string(lineChars);
        if(line.length() == 0)continue;
        if(line[0] == '.') break;
        int pid = stoi(line.substr(0,line.length()-4))-1;
        char rid = line[line.length()-3];
        char action = line[line.length()-1];

        if(action == 'U'){
            pair<char,bool> lock = pair<char,bool>(rid,false);
            processes[pid].locked.erase(lock);
            pair<int,bool> lockby = pair<int,bool>(pid,false);
            resources[rid].locked_by.erase(lockby);
            pair<char,bool> lock1 = pair<char,bool>(rid,true);
            processes[pid].locked.erase(lock1);
            pair<int,bool> lockby1 = pair<int,bool>(pid,true);
            resources[rid].locked_by.erase(lockby1);
        }
        if(action == 'W'){
            pair<char,bool> lock = pair<char,bool>(rid,false);
            processes[pid].locked.emplace(lock);
            pair<int,bool> lockby = pair<int,bool>(pid,false);
            resources[rid].locked_by.emplace(lockby);
        }
        if(action == 'R'){
            pair<char,bool> lock = pair<char,bool>(rid,true);
            processes[pid].locked.emplace(lock);
            pair<int,bool> lockby = pair<int,bool>(pid,true);
            resources[rid].locked_by.emplace(lockby);
        }
```

```
            if(checkDeadlocks()){
                    std::cout << "DEADLOCK\n";
                    for(int lockedPid : deadlocked){
                            std::cout << lockedPid+1 << " ";
                    }
                    std::cout << endl;
                    while(true){
                            cin.getline(lineChars,14);
                            string line = string(lineChars);
                            if(line[0] == '.') break;
                    }
                    return 0;
            }
    }

    std::cout << "NO DEADLOCK\n";
    int lockedCount = 0;
    for(pair<char,resource_t> r : resources){
            if(r.second.locked_by.size()>0)lockedCount++;
    }
    std::cout << lockedCount << endl;
    return 0;
}
```