Table of Contents

I – Số Nguyên – Kĩ thuật cài đặt cơ bản

1. Số Fibonacci

Số Fibonacci được xác định bởi công thức sau:

$F_0=0$

$F_1=1$

$F_n=F_{n-1}+F_{n-2}$ với $n\geq 2$.

Một số phần tử đầu tiên của dãy Fibonacci: 0,1,1,2,3,5,8,....

Tìm số Fibonacci thứ n.

Input

Một số nguyên dương duy nhất n (n≤1 000 000).

Output

Một số nguyên duy nhất là số Fibonacci thứ n (kết quả lấy phần dư cho 1 000 000 007).

Example

```
Input:
6


Output:
8
Input:
20
Output:
6765
```

```
#include<iostream>
using namespace std;
main(){
	long n;
	cin>>n;
	if(n == 0)
		cout<<0;
	else
		if(n == 1)
			cout<<1;
		else{
			long long x = 0,y = 1;
			for(long i=2;i<=n;i++){
				long long c = (x+y)%1000000007;
				x = y;
				y = c;
			}
			cout<<y;
		}
	}

```

2. Số nguyên tố

Một số được gọi là số nguyên tố nếu nó chỉ có 2 ước là 1 và chính nó. Số 0 và 1 không được coi là số nguyên tố.

Yêu cầu: Cho số n, hãy kiểm tra xem n có là số nguyên tố hay không.

Dữ liệu:

Một dòng duy nhất chứa số n (0<=n<=10^9)

Kết quả:

In ra “YES” nếu n là số nguyên tố, và “NO” trong trường hợp còn lại.

Ví dụ:

INPUT	OUTPUT
2	YES
INPUT	OUTPUT
4	NO

```
1. class HelloWorld{
2.     static int ngto(long a){
3.          if(a == 1||a==0) return 0;
4.          if(a == 2) return 1;
5.          if(a%2 == 0) return 0;
6.          for(int i=3;i<=(int)Math.sqrt(a);i+=2)
7.              if(a%i == 0) return 0;
8.          return 1;
9.      }
10.           public static void main(String[]args){
11.               Scanner nhap = new Scanner(System.in);
12.               long a;
13.               a = nhap.nextLong();
14.               if(ngto(a)==1)
15.                   System.out.print("YES\n");
16.               else
17.                   System.out.print("NO\n");
18.           }
19.       }
```

3. Số thuận nghịch

```
4. class phone {
5.          public static char sonao(char t){
6.              if(t == 'A' || t == 'B' || t == 'C' || t == 'a' || t
   == 'b' || t == 'c') return '2';
7.              if(t == 'D' || t == 'E' || t == 'F' || t == 'd' || t
   == 'e' || t == 'f') return '3';
8.              if(t == 'G' || t == 'H' || t == 'I' || t == 'g' || t
   == 'h' || t == 'i') return '4';
```

```
9.              if(t == 'J'  || t == 'K' || t == 'L' || t == 'j' || t
   == 'k' || t == 'l') return '5';
10.                 if(t == 'M'  || t == 'N' || t == 'O' || t == 'm'
   || t == 'n' || t == 'o') return '6';
11.                 if(t == 'P'  || t == 'Q' || t == 'R' || t == 'S'
   || t == 'p' || t == 'q' || t == 'r' || t == 's') return '7';
12.                 if(t == 'T'  || t == 'U' || t == 'V' || t == 't'
   || t == 'u' || t == 'v') return '8';
13.                 return '9';
14.             }
15.             public static String thanhso(String s){
16.                     StringBuilder b = new StringBuilder();
17.                     for(int i = 0; i < s.length(); i++)
18.                         b.append(sonao(s.charAt(i)));
19.                     String kq = b.toString();
20.                     return kq;
21.             }
22.             public static void xuly(String s){
23.                     s = thanhso(s);
24.
25.                     StringBuilder b = new StringBuilder(s);
26.                     b.reverse();
27.                     String s1 = b.toString();
28.                     if(s.equals(s1)) System.out.println("YES");
29.                     else    System.out.println("NO");
30.             }
31.             public static void main(String[] args) {
32.                     Scanner nhap = new Scanner(System.in);
33.                     int n = nhap.nextInt(), dem = 0;
34.                     String a[] = new String[1001];
35.                     for(int i = 0; i <= n; i++){
36.                         String temp = nhap.nextLine();
37.                         if(!temp.equals("")) a[dem++] = temp;
38.                     }
39.                     for(int i = 0; i < n; i++)
40.                         xuly(a[i]);
41.         }
```

4. Số chia may mắn- BCTEST11

Bờm rất yêu thích các số may mắn. Ta biết rằng một số gọi là *số may mắn* nếu biểu diễn thập phân của nó chỉ chứa các chữ số may mắn là 4 và 7. Ví dụ: Các số 47,744,4 là số may mắn còn 5,17,467 không phải là số may mắn.

Từ định nghĩa trên, Bờm định nghĩa ra khái niệm ***số chia may mắn.*** Một số là số chia may mắn nếu nó chia hết cho ít nhất một số may mắn. Ví dụ: 8,14,.. Bạn hãy giúp Bờm xác định xem số N có phải là ***số chia may mắn*** hay không?

Dữ liệu:

Một số nguyên duy nhất N (1≤N≤1000). .

Kết quả:

In ra "YES" nếu N là ***số chia may mắn***, hoặc "NO" nếu ngược lại.

Ví dụ:

INPUT	OUTPUT
47	YES
INPUT	OUTPUT
16	YES
INPUT	OUTPUT
78	NO

```
1. using namespace std;
2. int mm(int n){
3.     while(n>0){
4.           int i = n%10;
5.           if(i != 4 && i != 7) return 0;
6.           n /= 10;
7.     }
8.     return 1;
9. }
10.    void chiamm(int n){
11.          for(int i=4;i<=n;i+=3)
12.                if(mm(i) && n%i == 0){
13.                      cout<<"YES";
14.                      return;
15.                }
16.          cout<<"NO";
17.    }
18.    main(){
19.          int n;
20.          cin>>n;
21.          chiamm(n);
22.    }
23.
```

5. Số đối xứng

```
public class SoDoiXung {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        long test;
        test= sc.nextLong();
        while(test>0){
            test--;
            long n;
            n= sc.nextLong();
            doixung(n);

        }
    }
    static void doixung(long n){
        long m=0, a=n;
        while(n>0){
            m= m*10+ n%10;
            n/=10;
        }
        if(a==m)
            System.out.println("YES");
        else
            System.out.println("NO");
    }
}
```

6. Số nén tối giản

Ta gọi phép nén một số nguyên là tính tổng các chữ số của nó. Dễ thấy, sau một số phép nén, thì số còn lại chỉ có một chữ số và ko nén được nữa. Ta gọi số đó là số nén tối giản.

Ví dụ cho số 86. Sau phép nén thứ nhất ta đk: 8+6=14. Sau phép nén thứ 2: 1+4=5 => Số nén tối giản của 86 là 5.

Cho một số nguyên hãy tìm số nén tối giản của nó.

Input

Dòng đầu chứa số bộ test.

Mỗi dòng tiếp theo chứa 1 bộ test gồm 1 số nguyên dương. (<=10^9)

Mỗi dòng

Output

Mỗi dòng một số nén tối giản tương ứng.

Example

> **Input:**
>
> **3**
>
> **43**
>
> **111**
>
> **57871**
>
> **Output:**
> **7**
>
> **3**
>
> **1**

```
1. #include<iostream>
2. using namespace std;
3. int nen(long n){
4.       while(n>9){
5.             long i=n;
6.             int tong = 0;
7.             while(i>0){
8.                   tong += i%10;
9.                   i /= 10;
10.                  }
11.                  n = tong;
12.            }
```

```
13.             return n;
14.       }
15.       main(){
16.             long test;
17.             cin>>test;
18.             while(test--){
19.                   long n;
20.                   cin>>n;
21.                   cout<<nen(n)<<"\n";
22.             }
23.       }
```

20.

II- Xâu kí tự

1. Mã hóa xâu

Tí đang tìm hiểu một trong những kĩ thuật mật mã hóa đơn giản nhất. Với bản tin cần được mã hóa, phương pháp này được sẽ mã hóa thành một xâu với quy tắc như sau:
Xâu mã hóa chỉ lưu lại các kí tự chữ cái xuất hiện đầu tiên. Nói cách khác, các kí tự nào xuất hiện > 1 lần trở lên, sẽ bị xóa bỏ, chỉ giữ lại kí tự đầu tiên.
Giá trị mật mã của xâu được tính bằng tổng số lần xuất hiện của các kí tự có tần suất > 1 lần.
Các bạn hãy cùng Tí giải quyết bài toán này nhé!

Input

Dòng đầu tiên gồm số lượng test T (T <= 100).
T dòng tiếp theo, mỗi dòng gồm nhiều chuỗi các kí tự, có tổng độ dài không vượt quá 10^5.
Các kí tự là chữ cái in hoa hoặc chữ cái thường.

Output

Với mỗi test, in ra trên dòng số các giá trị mật mã của xâu, và xâu đó sau khi được mã hóa.

```
public class MaHoaXau {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      int n=Integer.parseInt(sc.nextLine());
      for(int i=0;i<n;i++){
         String s=sc.nextLine();
         s=s.replaceAll(" ", "");
         s=s.toLowerCase();
         String mh="";
         int m=s.length();
         int j;
         int dem[]=new int[m];
         for(j=0;j<m;j++){
```

```
            dem[j]=1;
        }
        for(j=0;j<m;j++){
            String t=s.substring(j, j+1);
            if(!mh.contains(t)){
                mh+=t;
            }else{
                dem[mh.indexOf(t)]++;
            }
        }
        int d=0;
        for(j=0;j<m;j++){
            if(dem[j]>1)d+=dem[j];
        }
        System.out.println(d+" "+mh);
    }

  }
}
```

2. Mật khẩu

Một xâu ký tự được gọi là mật khẩu "an toàn" nếu xâu có độ dài ít nhất bằng 6 và xâu chứa ít nhất một chữ cái in hoa , một chữ cái thường , một chữ số .
Ví dụ, 'a1B2C3', 'tinHoc6' là hai mật khẩu "an toàn". Còn 'a1B2C', 'a1b2c3', 'A1B2C3', 'tinHoc' đều không phải là mật khẩu "an toàn".
Một lần, Tí nhìn thấy một xâu S, chỉ gồm các loại ký tự: chữ cái in hoa, chữ cái thường và chữ số. Tí muốn tự kiểm tra khả năng đoán nhận mật khẩu bằng cách đếm xem có bao nhiêu cặp chỉ số (i, j) thỏa mãn điều kiện: $1 \le i < j \le length(S)$ và xâu con gồm các ký tự liên tiếp từ i đến j của S là mật khẩu "an toàn".
Cho xâu S, các hãy tính số lượng cặp chỉ số (i, j) thỏa mãn điều kiện nêu trên.

Input

Một dòng chứa xâu S có độ dài <= 10^6.

Output

In ra một số nguyên duy nhất là số cặp chỉ số (i, j) tìm được.

```
public class MatKhau {
   static boolean isContains(String s,char begin,char end){
      for(int i=0;i<s.length();i++){
         if(s.charAt(i)>=begin&&s.charAt(i)<=end){
            return true;
```

```
        }
      }
      return false;
   }
   static int Mk(String s){
      int dem=0;
      for(int i=0;i<s.length();i++){
         for(int j=i+6;j<=s.length();j++){
            String t=s.substring(i, j);
            if(isContains(t, 'A', 'Z')&&isContains(t, 'a', 'z')&&isContains(t, '0', '9')){
              dem++;
            }
         }
      }
      return dem;
   }
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      String s=sc.nextLine();
      System.out.println(Mk(s));

   }
}
```

3. Xâu cân bằng

Cho một xâu S gồm N kí tự (chỉ số các kí tự lần lượt là 0, 1, 2, ... N-1). Xâu R là xâu đảo ngược của xâu S. Xâu S được gọi là xâu cân bằng nếu như nó thỏa mãn điều kiện:

|S[i] – S[i-1]| = |R[i] – R[i-1]| với mọi i từ 1 à N-1, trong đó các giá trị S[i] được tính theo bảng mã ASCII.

Với các xâu S cho trước, các bạn hãy kiểm tra xem nó có là xâu cân bằng hay không?

Input

Dòng đầu tiên gồm số lượng test T (T <= 10).

T dòng tiếp theo, mỗi dòng gồm một xâu kí tự, gồm các chữ cái thường, độ dài không quá 10^5.

Với mỗi test, in ra “YES” nếu xâu đó là xâu cân bằng, in ra “NO” trong trường hợp ngược lại.

```
public class XauCanBang {
   static boolean canbang(String s){
      String st=new StringBuffer(s).reverse().toString();
      for(int i=s.length()-1;i>0;i--){
```

```
            if(Math.abs(s.charAt(i)-s.charAt(i-1))!=Math.abs(st.charAt(i)-st.charAt(i-1))){
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        for (int i = 0; i < n; i++) {
            String s = sc.nextLine();
            if(canbang(s))System.out.println("YES");
            else System.out.println("NO");
        }
    }
}
```

4. Xâu con lặp lại

Cho một xâu S có tối đa 100000 chữ cái. Các xâu con lặp lại được định nghĩa là một đoạn liên tiếp các ký tự được lặp lại ít nhất hai lần trong xâu S (các đoạn có thể chồng lấn lên nhau).
Ví dụ xâu “aabaab” có 5 xâu con lặp lại là “a”; “aa”, “aab”, “ab”, “b”.
Xâu “aaaaa” sẽ có 4 xâu con lặp lại là “a”, “aa”, “aaa”, “aaaa”.
Bài toán đặt ra là cho trước một xâu không quá 100000 ký tự chữ cái, hãy đếm số xâu con lặp lại của xâu đó.

Input
Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều test. Dòng đầu ghi số bộ dữ liệu (không quá 10).
Mỗi bộ dữ liệu ghi trên một dòng xâu S tương ứng.

Output
Với mỗi bộ dữ liệu, ghi trên một dòng số xâu con lặp lại đếm được. Kết quả được đảm bảo không quá giới hạn số nguyên 32 bít.

```
    static int SoXauCon(String s){
        int m=s.length();
        Stack st=new Stack();
        int dem=0;
        for(int i=m;i>1;i--){
            String k=s.substring(0, i-1);
            for(int j=i-1;j>0;j--){
                String t=s.substring(j, i);
```

```
            if(!st.contains(t)&&k.contains(t)){
              st.push(t);
              dem++;
            }
          }
        }
        return dem;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=Integer.parseInt(sc.nextLine());
        for(int i=0;i<n;i++){
            String s=sc.nextLine();
            System.out.println(SoXauCon(s));
        }

    }
}
```

5. Số nguyên tố(cơ bản)

Yêu cầu: Cho số n, hãy kiểm tra xem n có là số nguyên tố hay không.
Dữ liệu:
Một dòng duy nhất chứa số n (0<=n<=10^9)
Kết quả:
In ra "YES" nếu n là số nguyên tố, và "NO" trong trường hợp còn lại.
Ví dụ:

INPUT	OUTPUT
2	YES
INPUT	OUTPUT
4	NO

```
1. import java.util.Scanner;
2. class HelloWorld{
3.     static int ngto(long a){
4.           if(a == 1||a==0) return 0;
5.           if(a == 2) return 1;
6.           if(a%2 == 0) return 0;
7.           for(int i=3;i<=(int)Math.sqrt(a);i+=2)
8.               if(a%i == 0) return 0;
9.           return 1;
```

```
10.         }
11.         public static void main(String[]args){
12.             Scanner nhap = new Scanner(System.in);
13.             long a;
14.             a = nhap.nextLong();
15.             if(ngto(a)==1)
16.                 System.out.print("YES\n");
17.             else
18.                 System.out.print("NO\n");
19.         }
20.     }
```

6. Phân tích ra thừa số nguyên tố(cơ bản)

Cho số nguyên dương n (2<=n<=10^9) , hãy phân tích n ra thừa số nguyên tố.

Dữ liệu:

Một dòng duy nhất chứa số n.

Kết quả:

Mỗi dòng ghi một thừa số nguyên tố và số mũ tương ứng cách nhau bởi dấu cách.

Các thừa số nguyên tố in ra theo thứ tự tăng dần.

Ví dụ:

INPUT	OUTPUT
4	2 2
INPUT	OUTPUT
168	2 3 3 1 7 1

7. Đoạn Tăng

Cho dãy số nguyên A = (a_1, a_2, …, a_n). Hãy tìm một đoạn dài nhất gồm các phần tử **liên tiếp** trong dãy A có thứ tự không giảm

Quy ước: Đoạn chỉ gồm đúng 1 phần tử trong A cũng được coi là có thứ tự không giảm

Dữ liệu:

l Dòng 1 chứa số nguyên dương n ≤ 10^5

l Dòng 2 chứa n số nguyên a_1, a_2, ..., a_n ("i: $|a_i| \le 10^9$) cách nhau ít nhất một dấu cách

Kết quả:một số nguyên duy nhất là số phần tử trong đoạn tìm được

Ví dụ

INPUT	OUTPU

	T
11 88 99 11 22 22 33 11 66 33 44 77	4

```
#include<iostream>
using namespace std;
main(){
	int n;
	long x[100001];
	cin>>n;
	for(int i=1;i<=n;i++)
		cin>>x[i];
	int len=0;
		for(int i=1;i<=n;i++){
			int dem=1;
			for(int j=i+1;j<=n;j++)
				if(x[j-1] <= x[j]) dem++;
				else break;
			if(dem > len) len = dem;
		}
		cout<<len;
	}
```

8. Tách số

Cho đoạn văn có N dòng, mỗi dòng chỉ chứa các chữ số và chữ cái in thường trong bảng chữ cái tiếng Anh. Nhiệm vụ của bạn là tìm tất cả các số trong đoạn văn và in chúng ra thành một dãy số không giảm. Để cho đơn giản, các số có các chữ số 0 bắt đầu sẽ xóa bỏ các chữ số 0 ở đầu.

Các số được xác định duy nhất bằng cách quét qua đoạn văn và luôn lấy số lượng các chữ số lớn nhất có thể. Ví dụ: Các số 01a2b3456cde478 là 1, 2, 478, 3456.

Input

Dòng 1 chứa số nguyên N ($1 \le N \le 100$), là số dòng của đoạn văn.

N dòng tiếp theo, mỗi dòng chứa một chuỗi các chữ số và chữ cái in thường trong tiếng Anh. Mỗi dòng tối đa 100 kí tự.

Output

Gồm M dòng (M là số các số tìm được trong đoạn văn). Mỗi dòng chứa 1 số lấy từ đoạn văn. Các số được sắp xếp theo thứ tự không giảm. Chú ý: M luôn không quá 500.

Example

```
Input:
```

```
2
lo3za4
01
Output:
1
3
4
Input:
4
43silos0
zita002
le2sim
231233
Output:
0
2
2
43
231233
```

```java
	static Comparator<String> myCom = new Comparator<String>() {

		@Override
		public int compare(String a, String b) {
			if (a.length() == b.length())
				return a.compareTo(b);
			if (a.length() > b.length())
				return 1;
			return -1;
			}

		};

		public static void main(String[] args) {
			Scanner sc = new Scanner(System.in);
			ArrayList<String> a = new ArrayList<>();
			int n = Integer.parseInt(sc.nextLine());
			while (n-- > 0) {
				String s = sc.nextLine();
				for (int i = 0; i < s.length();) {
					char c = s.charAt(i);
```

```
                        if (c == '0' && i < s.length() - 1
                                    && !(s.charAt(i + 1) >= '0'
   && s.charAt(i + 1) <= '9')
                                    || (c == '0' && i ==
   s.length() - 1)) {
                              a.add("0");
                              i++;
                              continue;
                        }
                        if (c >= '1' && c <= '9') {
                              String r = "" + c;
                              int j = i + 1;
                              if (i == s.length() - 1) {
                                    a.add(r);
                                    break;
                              }
                              char b = s.charAt(j);
                              while (b >= '0' && b <= '9') {
                                    r += b;
                                    if (++j >= s.length())
                                          break;
                                    b = s.charAt(j);
                              }
                              a.add(r);
                              i = j;
                        } else
                              i++;
                  }
            }
            Collections.sort(a, myCom);
            for (String i : a)
                  System.out.println(i);
      }
}
```

9. Xâu thân thiện

Xâu T được gọi là thân thiện với xâu S nếu hoán vị các chữ cái trong xâu T tạo thành xâu S, ví dụ bab thân thiện với xâu abb còn xâu aab thì không.

Goku có một xâu S và một xâu T gồm các chữ cái in thường và kí tự '?'. Goku yêu cầu thánh phồng Saitama lấy ra từ xâu T một xâu con X gồm các kí tự liên tiếp trong T, với mỗi kí tự '?' trong X, thánh phồng có thể đổi thành một kí tự bất kì. Goku đố thánh phồng rằng có bao nhiêu xâu con X tách ra từ xâu T mà khi thay các kí tự '?' sẽ được một xâu thân thiện với xâu S.

Tất nhiên thánh phồng vẫn là thánh phồng. Anh chỉ cần "one punch" là giải được bài này rồi

Input

Dòng đầu gồm 1 xâu T gồm các chữ cái in thường và dấu '?' có độ dài không quá 10^5.

Dòng thứ 2 gồm 1 xâu S có độ dài không quá 10^5 gồm các chữ cái in thường.

Output

In ra một số nguyên duy nhất là số lượng xâu con trong T mà thân thiện với xâu S.

Example

Input:
?bba?
bba
Output:
3
Giải thích:
Xâu T có thể tách thành 3 xâu con ?bb (rồi chuyển thành abb), bba và ba? (rồi chuyển thành bab).

Bài làm:

```
class Main {
    public static boolean Matching(int[] OccOfTempT, int[] OccOfS) {
        int OccOfHoiCham = OccOfTempT[0];
        for (int i = 1; i < 27; i++) {
            if (OccOfTempT[i] > OccOfS[i]) {
                return false;
            } else {
                OccOfHoiCham -= OccOfS[i] - OccOfTempT[i]; } }
        if (OccOfHoiCham == 0) {
            return true; }
        return false; }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String t = sc.nextLine();
        String s = sc.nextLine();
        if (s.length() > t.length()) {
            System.out.println("0");
        } else {
            int[] OccOfTempT = new int[27];
            int[] OccOfS = new int[27];
            for (int i = 0; i < s.length(); i++) {
                OccOfS[s.charAt(i) - 96]++;
                char temp = t.charAt(i);
                if (temp == '?') {
                    OccOfTempT[0]++;
                } else {
                    OccOfTempT[temp - 96]++; } }
            int numberOfMatching = 0;
            if (Matching(OccOfTempT, OccOfS)) {
                numberOfMatching++; }
            int i = 1;
            while (i <= t.length() - s.length()) {
                char temp = t.charAt(i - 1);
                if (temp == '?') {
                    OccOfTempT[0]--;
                } else {
                    OccOfTempT[temp - 96]--; }
                temp = t.charAt(i + s.length() - 1);
                if (temp == '?') {
                    OccOfTempT[0]++;
                } else {
                    OccOfTempT[temp - 96]++; }
```

```
            if (Matching(OccOfTempT, OccOfS)) {
               numberOfMatching++; }
            i++;}
         System.out.println(numberOfMatching);}}}
```

10. SỐ ĐẸP 1

Một số được coi là đẹp nếu nó là số thuận nghịch, tổng chữ số là số nguyên tố và tất cả các chữ số đều lẻ. Bài toán đặt ra là đếm xem trong một đoạn giữa hai số nguyên cho trước có bao nhiêu số đẹp như vậy.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống. Các số đều không vượt quá 9 chữ số.

Kết quả:

Với mỗi bộ test viết ra số lượng các số thuần nguyên tố tương ứng.

Ví dụ

Input

3

23 199

2345 6789

222222 99999999

Output

4 0

311

Bài làm:

```
public class SD{
   static boolean thuanNghich(long n) {
      long k = n, p = 0;
      while (k != 0) {
         p = p * 10 + k % 10;
         k /= 10;}
      return (n == p);}
   static boolean isSNT(long n){
      if (n == 0 || n == 1 || n % 2 == 0) {
         return false;}
      long ll = (long) Math.sqrt(n);
      for (long i = 3; i <= ll; i+=2) {
         if (n % i == 0) {
            return false;}}
      return true;}
   static boolean laNguyenToVaSole(long n){
      long sum = 0;
      long cs;
      do {
         cs = n % 10;
         if (cs % 2 == 0) {
```

```
            return false;}
        sum += cs;
        n /= 10;
    } while (n > 0);
    return isSNT(sum);}
static int dem(long a, long b){
    int dem = 0;
    if (a % 2 == 0) {
        a += 1;}
    for (long i = a; i <= b; i += 2) {
        if (thuanNghich(i) && laNguyenToVaSole(i)) {
            dem++;}}
    return dem;}
public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int soBoTest = Integer.parseInt(sc.nextLine());
    long[] a = new long[soBoTest];
    long[] b = new long[soBoTest];
    String input;
    for (int i = 0; i < soBoTest; i++) {
        input = sc.nextLine();
        a[i] = Long.parseLong(input.split(" ")[0]);
        b[i] = Long.parseLong(input.split(" ")[1]);}
    for (int i = 0; i < soBoTest; i++) {
        System.out.println(dem(a[i], b[i]));}}}
```

11. **Số đẹp 2**.

Một số được coi là đẹp nếu nếu nó có tính chất thuận nghịch và tổng chữ số chia hết cho 10. Bài toán đặt ra là cho trước số chữ số. Hãy đếm xem có bao nhiêu số đẹp với số chữ số như vậy.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng số chữ số tương ứng cần kiểm tra (lớn hơn 1 và nhỏ hơn 10)

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra số lượng số đẹp tương ứng

Ví dụ

Input

2

2

5

Output

1

90

Bài làm:

```
public class SD2{
   static boolean thuanNghich(long n)    {
      long k = n, p = 0;
      while (k != 0) {
         p = p * 10 + k % 10;
         k /= 10;}
      return (n == p);}
   static boolean isChiaHetCho10(long n){
      if (n == 2) {
         return true;}
      long sum = 0;
      do {
         sum += n % 10;
         n /= 10;
      } while (n > 0);
      return ((sum % 10) == 0);}
   static int dem(long a){
      int dem = 0;
      long b = (long) Math.pow(10, a);
      for (long i = (long) Math.pow(10, a - 1); i < b; i++) {
         if (thuanNghich(i) && isChiaHetCho10(i)) {
            dem++;}}
      return dem;}
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int soBoTest = Integer.parseInt(sc.nextLine());
      long[] a = new long[soBoTest];
      String input;
      for (int i = 0; i < soBoTest; i++) {
         input = sc.nextLine();
         a[i] = Long.parseLong(input);}
      for (int i = 0; i < soBoTest; i++) {
         System.out.println(dem(a[i]));}}}
```

12. Số thuần nguyên tố.

Một số được coi là thuần nguyên tố nếu nó là số nguyên tố, tất cả các chữ số là nguyên tố và tổng chữ số của nó cũng là một số nguyên tố. Bài toán đặt ra là đếm xem trong một đoạn giữa hai số nguyên cho trước có bao nhiêu số thuần nguyên tố.

Dữ liệu vào:Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống. Các số đều không vượt quá 9 chữ số.

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra số lượng các số thuần nguyên tố tương ứng.

Ví dụ
Input
11
2
23 199
2345 6789
Ouput
1
15
Bài làm:

```
public class TNT{
   static boolean isSNT(long n){
      if (n == 2) {
         return true; }
      if (n == 0 || n == 1 || n % 2 == 0) {
         return false;}
      long ll = (long) Math.sqrt(n);
      for (long i = 3; i <= ll; i += 2) {
         if (n % i == 0) {
            return false;}}
      return true;}
   static boolean laNguyenTo(long n){
      long sum = 0;
      long cs;
      do {
         cs = n % 10;
         if (isSNT(cs)) {
            sum += cs;
            n /= 10;
         } else {
            return false;}
      } while (n > 0);
      return isSNT(sum);}
   static int dem(long a, long b){
      int dem = 0;
      if (a % 2 == 0) {
         a += 1;}
      for (long i = a; i <= b; i += 2) {
         if (isSNT(i) && laNguyenTo(i)) {
            dem++;}}
      return dem;}
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int soBoTest = Integer.parseInt(sc.nextLine());
```

```
        long[] a = new long[soBoTest];
        long[] b = new long[soBoTest];
        String input;
        for (int i = 0; i < soBoTest; i++) {
           input = sc.nextLine();
           a[i] = Long.parseLong(input.split(" ")[0]);
           b[i] = Long.parseLong(input.split(" ")[1]); }
        for (int i = 0; i < soBoTest; i++) {
           System.out.println(dem(a[i], b[i]));}}}
```

13. SỐ xuất hiện nhiều nhất trong dãy.

Cho một dãy số nguyên dương không quá 100 phần tử, các giá trị trong dãy không quá 30000. Hãy xác định xem số nào là số xuất hiện nhiều lần nhất trong dãy. Chú ý: trong trường hợp nhiều số khác nhau cùng xuất hiện số lần bằng nhau và là lớn nhất thì in ra tất cả các số đó theo thứ tự xuất hiện trong dãy ban đầu.

Dữ liệu vào:

Dòng đầu là số bộ test, không quá 20.

Mỗi bộ test gồm hai dòng. Dòng đầu ghi số phần tử của dãy, dòng tiếp theo ghi các phần tử của dãy.

Kết quả: Ghi ra màn hình

Với mỗi bộ test, đưa ra số xuất hiện nhiều lần nhất trong dãy đã cho.

Ví dụ:

Input

2

10

1 2 3 1 2 3 1 2 3 1

10

1 2 3 4 5 6 7 8 9 0

Output

1

1 2 3 4 5 6 7 8 9 0

Bài làm:

```
public class So_xuat_hien_nhieu_nhat {
        public static int A[]=new int[30001];
        public static int B[];
        public static void main(String[]args){
                Scanner inp=new Scanner(System.in);
                int m = Integer.parseInt(inp.nextLine());
                for(int testcase=0;testcase<m;testcase++){
                        A=new int[30001];
                        int length = Integer.parseInt(inp.nextLine());
                        B=new int[length];
                        String temp=inp.nextLine();
                        String[] listS = temp.split(" ");
```

```
for(int i=0;i<length;i++){
        B[i] = Integer.parseInt(listS[i]);
        A[B[i]] ++; }
int max=0;
for(int i=0;i<B.length;i++){
        if(A[B[i]]>max){
                max=A[B[i]]; }}
System.out.println("testcase "+testcase);
for(int i=0;i<B.length;i++){
        if(A[B[i]]==max){
                System.out.print(B[i]+" ");
                A[B[i]]=0; }}
System.out.println();}}}
```

14. Thuận nghịch dài nhất.

Cho một file văn bản bất kỳ. Hãy tìm ra từ thỏa mãn tính chất thuận nghịch có độ dài lớn nhất trong file đó và cho biết từ đó xuất hiện bao nhiêu lần. Nếu có nhiều từ cùng có độ dài lớn nhất thì in ra tất cả các từ đó theo thứ tự xuất hiện trong file ban đầu.

Dữ liệu vào: File C.IN

Gồm một đoạn văn bản bất kỳ. Không quá 1000 từ.

Kết quả (ghi ra màn hình):

- Ghi ra trên một dòng từ thuận nghịch có độ dài lớn nhất và số lần xuất hiện của nó.
- Nếu có nhiều từ cùng có độ dài lớn nhất thì các từ được liệt kê theo thứ tự xuất hiện ban đầu.

Ví dụ:

C.IN

AAA BAABA HDHDH ACBSD SRGTDH DDDDS

DUAHD AAA AD DA HDHDH AAA AAA AAA AAA

DDDAS HDHDH HDH AAA AAA AAA AAA AAA

AAA AAA AAA

DHKFKH DHDHDD HDHDHD DDDHHH HHHDDD

TDTD

KẾT QUẢ

HDHDH 3

Bài làm:

```
public class Thuan_nghich_dai_nhat {
    public static boolean flag[];
    public static int count[];
    public static boolean kiem_tra_thuan_ngich(String text){
            StringBuilder sb=new StringBuilder(text);
            String temp=String.valueOf(sb.reverse());
            if(temp.equals(text)){
                    return true;
            }else{
                    return false; }}
```

```
public static void main(String[]args){
        Scanner inp=new Scanner(System.in);
        String document = inp.nextLine();
        String[] listS = document.split(" ");
        flag=new boolean[listS.length];
        count=new int[listS.length];
        for(int i=0;i<listS.length;i++){
                flag[i]=true;
                count[i]=1;}
        for(int i=0;i<listS.length;i++){
                if(flag[i] && kiem_tra_thuan_ngich(listS[i])){
                        for(int j=0;j<listS.length;j++){
                                if(listS[i].equals(listS[j]) && i!=j ){
                                        flag[j]=false;
                                        count[i]++;}}
                }else{
                        flag[i]=false;}}
        int max=0;
        for(int i=0;i<listS.length;i++){
                if(flag[i]){
                        if(listS[i].length()>max){
                                max=listS[i].length();}}}
        for(int i=0;i<listS.length;i++){
                if(listS[i].length()==max && flag[i]==true){
                        System.out.println(listS[i]);}}}}
```

15. **Bài toán tách từ**.

Cho file dữ liệu văn bản data.in. Hãy tìm tập các từ và số lần xuất hiện mỗi từ trong file?

data.in

AB AC AD AE AF

AB AC AD AE AF

ketqua.out

5

AB 2

AC 2

AD 2

AE 2

AF 2

Bài làm:

```
public class Xau {
        public static void main(String[] args) throws FileNotFoundException, IOException {
    FileInputStream fis = new FileInputStream("input.txt");
```

```
        Scanner in = new Scanner(fis);
        FileWriter out = null;
        //ChuanHoaXau chx = new ChuanHoaXau();
        try {
            out = new FileWriter("output.txt");
            //int n = Integer.parseInt(in.nextLine());
            String data = "";
            ArrayList<String> words = new ArrayList<>();
            // bat dau doc file luu vao mang words
            while (in.hasNext()) {
                data = in.nextLine();
                data = data.replaceAll("\\s+", " ");
                String[] temp = data.split("\\s",0);
                for(int i=0;i<temp.length;i++){
                    words.add(temp[i]); } }
            int[] check = new int[words.size()];
            for (int i = 0; i < words.size(); i++) {
                check[i]=-1; }
            // dem trong maang word
            for (int i=0;i<words.size();i++) {
                int dem=1;
                for (int j = i+1; j < words.size(); j++) {
                    if (words.get(i).equals(words.get(j))){
                        dem++;
                        check[j]=0; }}
                if(check[i]!=0){
                    check[i]=dem; }}
            for (int i = 0; i < words.size(); i++) {
                if(check[i]!=0){
                    out.write(words.get(i)+"-"+check[i]);}}
        } finally {
            if (in != null) {
                in.close(); }
            if (out != null) {
                out.close();}}}}
```

Cách 2:

```
public class DocTu {
    static HashMap<String, Integer> ds = new HashMap<String, Integer>(){
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(new File("test.txt"));
            while (sc.hasNext()) {
```

```
        String s = sc.next();
        s = s.trim();
        if(ds.get(s)!=null){
           int soluong = ds.get(s);
           ds.replace(s, soluong+1);
        }else{
           ds.put(s,1);}}
     for (Map.Entry entry : ds.entrySet()) {
        System.out.println(entry.getKey() + " : " + entry.getValue());}
  } catch (Exception e) {}}
```

16. BCATM2 ATM2

Một máy ATM hiện có n (n <= 30) tờ tiền có giá trị t[1], t[2], …, t[n]. Hãy tìm cách trả ít tờ nhất với số tiền đúng bằng S.

Input

Dòng đầu tiên gồm 2 số nguyên n và S (S <= 10^9)

Dòng thứ hai chứa n số nguyên t[1], t[2], …, t[n] (t[i] <= 10^9)

Output

Số tờ tiền ít nhất phải trả.

Example

Input:

```
3 5
1 4 5
```

Output:

```
1
```

Bài làm:

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n,s;
        n=sc.nextInt();
        s=sc.nextInt();
        int []a=new int[n];
        for (int i=0;i<n;i++){
           a[i]=sc.nextInt(); }
        Arrays.sort(a);
        int i=a.length-1;
        int num=0;
        while(i>=0&&s!=0){
           if (s>=a[i]){
            num+=s/a[i];
            s%=a[i]; }
           i--; }
        System.out.println(num); } }
```

17. P167PROF Trò chơi trên bảng

Cho bảng số A gồm n hàng và n cột. Các hàng được đánh số từ 1 đến n, từ trên xuống dưới, các cột của lưới được đánh số từ 1 đến n, từ trái sang phải. Ô giao của hàng i và cột j gọi là ô (i, j) và được điền một số nguyên a_{ij} có giá trị tuyệt đối không vượt quá 10^9.

Xét trò chơi đối kháng giữa hai người trên bảng như sau: Trò chơi diễn ra trong n lượt đi, mỗi lượt người thứ nhất chọn một hàng, người thứ hai chọn một cột. Giả sử, tại một lượt đi, nếu người chơi thứ nhất chọn hàng i, người chơi thứ hai chọn cột j, khi đó người thứ nhất được cộng a_{ij} điểm, người thứ hai được cộng $-a_{ij}$ điểm. Sau lượt đi đó, bảng bị xóa hàng và cột mà hai người chơi vừa chọn. Người có điểm càng cao càng thể hiện sự thông minh của mình.

Yêu cầu: Cho bảng số A và biết cả hai người đều chơi tối ưu, hãy tính điểm lớn nhất mà người thứ nhất có thể đạt được.

Input

Dòng đầu tiên ghi K là số bộ test. Tiếp theo là K nhóm dòng, mỗi nhóm là một bộ test có cấu trúc như sau:

- Dòng đầu của nhóm ghi số n *(n ≤ 16);*
- n dòng sau, mỗi dòng chứa n số nguyên mô tả bảng số.

Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

Output

Gồm K dòng, mỗi dòng một số nguyên là điểm lớn nhất có thể của người thứ nhất tương ứng với dữ liệu vào.

Example

Input:

```
3
2
10 10
-5 -5
2
10 -5
10 -5
2
10 -5
-5 10
```

Output:

```
5
5
-10
```

Bài làm:

```
public class P167PROF {
    static int n;
    static int mark[] = new int[17];
    static long sum, ans, min_cost;
    static long min_row[] = new long[17];
    static long x[][] = new long[17][17];
```

```
static void reset() {
    ans = Long.MAX_VALUE;
    for (int i = 0; i < 17; i++) {
        mark[i] = 0; }
    sum = 0; }
static void get_min_row() {
    for (int i = 0; i < n; i++) {
        min_row[i] = Integer.MAX_VALUE;
        for (int j = 0; j < n; j++) {
            min_row[i] = Math.min(min_row[i], x[i][j]); } }
    for (int i = 1; i < n; i++) {
        min_row[i] += min_row[i - 1]; } }
static void Try(int step) {
    for (int j = 0; j < n; j++) {
        if (mark[j] == 0) {
            sum += x[step][j];
            mark[j] = 1;
            if (step == n - 1) {
                ans = Math.min(ans, sum);
            } else {
                min_cost = sum + min_row[n - 1] - min_row[step];
                if (sum + min_row[n - 1] - min_row[step] < ans) {
                    Try(step + 1); } }
            mark[j] = 0;
            sum -= x[step][j]; } } }
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int test = sc.nextInt();
    while (test > 0) {
        reset(); // reset the variables
        n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            min_row[i] = Integer.MAX_VALUE;
            for (int j = 0; j < n; j++) {
                x[i][j] =  sc.nextLong(); } }
        get_min_row();
        Try(0);
        System.out.println(ans);
        test--; } } }
```

18. BCCAITUI Cái túi

Trong siêu thị có n gói hàng (n <= 100), gói hàng thứ i có trọng lượng là W_i <= 100 và giá trị V_i <= 100. Một Tên trộm đột nhập vào siêu thị, sức của tên trộm không thể mang được trọng lượng vượt quá M (M <= 100). Hỏi tên trộm sẽ lấy đi những gói hàng nào để được tổng giá trị lớn nhất.

Input

Dòng đầu tiên gồm 2 số nguyên n và M (n <= 100, M <= 100)

Trên n dòng tiếp theo, mỗi dòng chứa 2 số nguyên W_i và V_i (W_i, V_i <= 100) lần lượt là trọng lượng và giá trị của gói hàng thứ i.

Output

Giá trị lớn nhất tên trộm lấy được.

Example

> **Input:**
> 3 4
> 1 4
> 2 5
> 3 6
> **Output:**
> 10

Bài làm:

```
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n,m;
      n=sc.nextInt();
      m=sc.nextInt();
      int []w=new int[n+1];
      int []v=new int[n+1];
      for (int i=1;i<=n;i++){
         w[i]=sc.nextInt();
         v[i]=sc.nextInt(); }
      int [][]A=new int [n+1][m+1];
      for (int i=1;i<=n;i++)
         for (int j=1;j<=m;j++){
            if (w[i]>j){
               A[i][j]=A[i-1][j];
            } else {
               A[i][j]=Math.max(A[i-1][j], A[i-1][j-w[i]]+v[i]); } }
System.out.println(A[n][m]); } }
```

19. PTIT122C Giai thừa 2

Tìm số lần xuất hiện của chữ số x trong số n! (n giai thừa).

Giới hạn:

- 1<=n<=365
- x là các chữ số từ 0 đến 9

Input

- Dòng đầu là số bộ test T (<=20)
- Sau đó là T dòng, mỗi dòng gồm 2 số nguyên cách nhau bởi dấu cách lần lượt là n và x.

Output

- Mỗi bộ test in trên một dòng đáp án.

Example

Input:
2
5 2
7 0
Output:
1
2

Bài làm:

```
public class PTIT122C {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int a = sc.nextInt();
            int c = sc.nextInt();
            int res[] = new int[300];
            res[0] = 1;
            int count = 0;
            int res_size = 1;
            for (int x = 2; x <= a; x++) {
                res_size = multiply(x, res, res_size); }
            for (int j = res_size - 1; j >= 0; j--) {
                if (res[j] == c) {
                    count++; } }
            System.out.println(count); } }
    static int multiply(int x, int res[], int res_size) {
        int carry = 0;  // Initialize carry

        for (int i = 0; i < res_size; i++) {
            int prod = res[i] * x + carry;
            res[i] = prod % 10;  // Store last digit of 'prod' in res[]
            carry = prod / 10;  }  // Put rest in carry
        while (carry != 0) {
            res[res_size] = carry % 10;
            carry = carry / 10;
            res_size++; }
        return res_size; } }
```

20. PTIT121I Lặp lại các ký tự

Với một xâu ký tự S, và một số nguyên R, hãy tạo ra một xâu T bằng cách mỗi ký tự trong S được lặp lại R lần. Các ký tự trong S có thể nằm trong dãy:
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ$%*+-./:

Input

- Dòng 1 ghi số bộ test (không quá 1000)
- Mỗi bộ test gồm số thứ tự, một dấu cách, tiếp theo là số 1<=R<=8, một dấu cách, rồi đến xâu ký tự S (không quá 20 ký tự).

Output

Với mỗi bộ test ghi trên một dòng số thứ tự bộ test, một dấu cách rồi đến xâu T.

Example

> **Input:**
> 2
> 1 3 ABC
> 2 5 /HTP
> **Output:**
> 1 AAABBBCCC
> 2 /////HHHHHTTTTTPPPPP

Bài làm:

```
public class PTIT121I {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());
      for (int i = 0; i < n; i++) {
         String s[] = sc.nextLine().split(" ");
         int mul = Integer.parseInt(s[1]);
         String txt = s[2];
         String out = "";
         for (int j = 0; j < txt.length(); j++) {
            for (int k = 0; k < mul; k++) {
               out+=txt.charAt(j); } }
System.out.println(s[0]+" "+out); } } }
```

21. Tìm kiếm file

Máy tính của Blabla bị hỏng nên tìm kiếm file ra kết quả sai, khi nhập một từ khóa tìm kiếm vào thì kết quả tìm kiếm bị sai rất nhiều khiên Blabla tức tối.

Một từ khóa tìm kiếm là một xâu chỉ chứa các ký tự tiếng Anh viết thường, và chứa duy nhất một dấu sao (*), ký tự * không nằm ở đầu cũng như ở cuối xâu. Một file khớp với một từ khóa tìm kiếm khi thay dấu * bằng một xâu bất kỳ (kể cả xâu rỗng) bằng nhau. Ví dụ “a*d” khớp với các file “abcd”, “abd”, “ad”, không khớp với “bcd”.

Hãy viết một chương trình giúp Blabla kiểm tra xem một file nào đó có khớp với xâu tìm kiếm không.

Input

Dòng đầu tiên chứa số tự nhiên n là số file (1 <= n < 100).

Dòng thứ hai chứa xâu tìm kiếm độ dài nhỏ hơn 100.

n dòng tiếp theo mỗi dòng là tên của một file.

Output

In ra trên mỗi dòng tương ứng “DA” nếu file đó khớp, ngược lại in “NE”.

Example

```
Input:
3
a*d
abcd
anestonestod
facebook

Output:
DA
DA
NE
```

```java
class Main2 {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int n = Integer.parseInt(sc.nextLine());
		String s = sc.nextLine();
		String a = "", b = "";
		int k = 0;
		for (int i = 0; i < s.length(); i++) {
			if (s.charAt(i) == '*') {
				k = i;
				break;
			}
			a += s.charAt(i);
		}
		for (int i = k + 1; i < s.length(); i++) {
			b += s.charAt(i);
		}
	//		System.out.println(a + " " + b);
		for (int i = 0; i < n; i++) {
			String t = sc.nextLine();

			String c = "", d = "";
			for (int j = 0; j < a.length()&&j<t.length(); j++) {
				c += t.charAt(j);
```

```
                        }
                        for (int j = t.length() - 1; j >=
   t.length() - b.length()&&j>=0; j--) {
                            d = t.charAt(j) + d;
                        }
      //                System.out.println(c + " " + d);

        if(c.equals(a)&&b.equals(d))System.out.println("DA");
                        else System.out.println("NE");
                    }
                }

        }

```

22. Chuẩn hóa xâu

Cho một xâu S gồm các ký tự hoa, thường.

Bạn hãy chuẩn hóa xâu S theo quy tắc, xâu kết quả chỉ gồm chữ hoa hoặc thường, là chữ thường khi số ký tự thường lớn hơn hoặc bằng số ký tự hoa của S và ngược lại.

Input

Xâu S có độ dài không quá 100 ký tự.

Output

Xâu S có độ dài không quá 100 ký tự.

Example

> **Input:**
>
> loWeR
>
> **Output:**
>
> lower

```
class CF {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s=sc.nextLine();
        int lower=0, upper=0;
        for (int i = 0; i <s.length() ; i++) {
            char c=s.charAt(i);
            if(c>='a'&&c<='z')lower++;
            else if(c>='A'&&c<='Z')upper++;
              }
```

```
            if(lower>=upper)
   System.out.println(s.toLowerCase());
            else System.out.println(s.toUpperCase());
        }
    }
```

23. Bóng đá

Tèo đi học và bỏ lỡ không xem được trận tranh huy chương đồng của seagame giữa Việt Nam và Indonesia. Cậu tiếc ngẩn, tiếc ngơ, cũng may thay bạn cậu là Tí xem được trận này và mỗi khi có bàn thắng cậu lại ghi tên đội ghi bàn vào một tờ giấy, với tờ giấy Tèo đã biết được đội thắng. Giờ đây Tỉ giả sử với những tờ giấy bất kì như vậy, cậu cần biết ngay đội thắng mà không cần nhọc công xem lại từ đầu đến cuối, các bọn giúp tí giảm bớt công sức nhé.

Input

Dòng đầu tiên là số nguyên n(1 <= n <= 100) là số bàn thắng của trận đấu.

n dòng tiếp theo, mỗi dòng là tên đội bóng ghi bàn.(Tên đội viết hoa, không quá 10 kí tự)

Đầu vào đảm bảo chứa không quá 2 đội và trận đấu không kết thúc với tỉ số hòa.

Output

Tên đội thắng

Example

> **Input:**
>
> 5
>
> Vietnam
>
> Vietnam
>
> Vietnam
>
> Vietnam
>
> Vietnam
>
> **Output:**
> Vietnam

```
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int d = 0, n = sc.nextInt();
        for (int i = 0; i < n; i++) {
```

```
6.                  String s = sc.next();
7.                  if (s.equals("Vietnam"))
8.                        d++;
9.                  else
10.                             d--;
11.                 }
12.                 if (d > 0)
13.                       System.out.println("Vietnam");
14.                 else
15.                       System.out.println("Indonesia");
16.           }
17.     }
```

24. Biến đổi chuỗi

Cho chuỗi S gồm các chữ cái Latin hoặc là in thường hoặc là in hoa. Yêu cầu bạn hãy viết chương trình thực hiện các nhiệm vụ sau với chuỗi S đã cho:

- Xóa tất các các nguyên âm.
- Chèn kí tự "." Vào trước mỗi phụ âm.
- Thay thế tất cả các chữ cái in hoa bằng chữ cái in thường tương ứng.

Nguyên âm là các chữ cái: "A", "O", "Y", "E", "U", "I","a", "o", "y", "e", "u", "i".

Dữ liệu:

Một chuỗi S duy nhất gồm các chữ cái Latin in thường và in hoa có độ dài từ 1 tới 100 chữ cái.

Kết quả:

Chuỗi S sau khi thực hiện các nhiệm vụ ở đề bài.

Ví dụ:

INPUT	OUTPUT
Hocviencnbcvt	.h.c.v.n.c.n.b.c.v.t

INPUT	OUTPUT
PTITOJ	.p.t.t.j

```
1. using namespace std;
2. main(){
3.       char s[100],kq[100];
```

```
4.       cin>>s;
5.       int n = strlen(s), dem = 0;
6.       for(int i=0;i<n;i++){
7.            if(s[i]=='A'||s[i]=='I'||s[i]=='U'||s[i]=='E'||
   s[i]=='O'||s[i]=='Y'||s[i]=='a'||s[i]=='i'||s[i]=='u'||
   s[i]=='e'||s[i]=='o'||s[i]=='y')
8.                 continue;
9.            kq[dem] = '.';
10.                dem++;
11.                if(s[i]>'A'&&s[i]<='Z') kq[dem] = s[i]+32;
12.                else kq[dem] = s[i];
13.                dem++;
14.           }
15.           kq[dem] = '\0';
16.           cout<<kq;
17.      }
```

25. Sâu Con

1 con sâu chiều dài n sẽ được biểu diễn dưới dạng 1 xâu có n chữ cái: s[0], s[1], ..., s[n-1].

Một con sâu con của 1 con sâu sẽ được biểu diễn dưới dạng xâu: s[p1], s[p2], ..., s[pk] (1 <= p1 < p2 < ... < pk <= n-1).

Cho 1 xâu là biểu diễn của 1 con sâu, tìm biểu diễn của con sâu con thứ tự cao nhất trong từ điển,

1 xâu x (x[0], x[1], ..., x[n-1]) có thứ tự từ điển cao hơn xâu y (y[0], y[1], ..., y[m-1]) khi và chỉ khi n > m và x[0]=y[0], x[1]=y[1], ..., x[m-1]=y[m-1]; hoặc x[0]=y[0], x[1]=y[1], ... x[r] = y[r] và x[r+1] > y[r+1]. Các ký tự được so sánh theo bảng mã ASCII.

Input

Dòng duy nhất bao gồm 1 xâu ký tự có chiều dài không quá 10^5.

Output

In ra trên 1 dòng duy nhất xâu ký tự là kết quả của bài toán

Example

Test 1:

Input:

ababba

Output:

bbba

Test 2:

Input:

laptrinhtutraitim

Output:

uttm

```
1. using namespace std;
2. string daoChuoi(string s){
3.       string t = "";
4.       for(int i = s.size() - 1; i >= 0; i--){
5.             t += s[i];
6.       }
7.       return t;
8. }
9. main(){
10.            string s, t = "";
11.            cin>>s;
12.            stack<char> S;
13.            for(int i = 0; i < s.size(); i++){
14.                  while(!S.empty() && s[i] > S.top()){
15.                        S.pop();
16.                  }
17.                  S.push(s[i]);
18.            }
19.            while(!S.empty()){
20.                  t += S.top();
21.                  S.pop();
22.            }
23.            cout<<daoChuoi(t);
24.      }
```

II- Xử lý Mảng và Ma Trận

1. Đếm số lần xuất hiện

```
Public class DemSoLanXuatHien {
   static void dem(int a[],int b[],int n){
      for(int i=0;i<n;i++)b[i]=0;
      for(int i=0;i<n;i++){
         if(b[i]==0){
```

```
            b[i]=1;
            for(int j=i+1;j<n;j++){
              if(a[j]==a[i]){
                b[i]++;
                b[j]=-1;
              }
            }
          }
        }
        for(int i=0;i<n;i++){
          if(b[i]>0)System.out.println(a[i]+ " "+b[i]);
        }
    }
    public static void main(String[] args) {
        int n=8;
        int a[]={3,5,7,5,3,7,20,7};
        int b[]=new int[n];
        dem(a,b,n);
    }
```

2. Nhân hai ma trận

3. Lũy Thừa

Lũy thừa bậc n của a bằng tích của n thừa số bằng nhau, mỗi thừa số có giá trị bằng a.

Cho trước 2 số nguyên a và b, các bạn hãy viết chương trình tính giá trị lũy thừa a^b.

Input

Gồm nhiều test, mỗi test ghi trên 1 dòng, gồm 2 số nguyên không âm a và b (a <= 10^9 và b <= 10^18).

Input kết thúc bởi 2 số 0.

Output

Với mỗi test, ghi ra trên một dòng kết quả phép tính lũy thừa a^b được lấy dư theo 1000 000 007.

Example

> **Input:**
>
> ```
> 2 3
> 2 4
> 3 2
> 0 0
> ```

Output:

```
8
16
9
```

```
1. #include <iostream>
2. using namespace std;
3. int ok(long a,long long b){
4.    if(a==0&&b==0) return 0;
5.    return 1;
6. }
7. main(){
8.    long a,c=1000000007;long long b;
9.    cin>>a>>b;
10.         while(ok(a,b)){
11.               int bi[100],k=0;
12.               int tam[100];
13.               while(b>0){
14.                     k++;
15.                     tam[k]=b%2;
16.                     b/=2;
17.               }
18.               for(int i=1;i<=k;i++)
19.                     bi[i]=tam[k+1-i];
20.               long long p=1;
21.               for(int i=1;i<=k;i++){
22.                     p=(p*p)%c;
23.                     if(bi[i]==1){
24.                           p=(p*a)%c;
25.                     }
26.               }
27.               cout<<p<<"\n";
28.               cin>>a>>b;
29.         }
30.   }
```

4. Nhặt táo

Mirko vừa tìm thấy 1 trò chơi điện tử cũ. Màn hình của game chia thành N cột. Ở dưới của màn hình , có 1 con thuyền chứa trong M cột (M<N). Người chơi có thể di chuyển thuyền sang trái hoặc phải, nhưng phải trong phạm vi của màn hình. Ban đầu, thuyền ở M cột bên trái nhất của màn hình.

Những quả táo sẽ rơi từ phía trên của màn hình. Mỗi quả bắt đầu rơi ở đầu trên của một cột, và rơi xuống phía dưới của màn hình. Quả tiếp theo rơi sau khi quả trước đó chạm đáy.

Nhiệm vụ của bạn là tìm cách di chuyển ngắn nhất để có thể lấy tất cả trái táo.

Dữ liệu:

Dòng đầu chứa 2 số nguyên N và M (1<=M<=N<=10).

Dòng 2 chứa số nguyên J (1<=J<=20), số trái táo rơi.

J dòng sau là thứ tự các cột của các quả táo sẽ rơi.

Kết quả:

Dòng chứa số nguyên duy nhất là số di chuyển bé nhất để lấy tất cả trái táo.

Example

```
Input:
5 1
3
1
5
3

Output:
6
```

```
1. public class Nhattao {
2.
3.     /**
4.      * @param args the command line arguments
5.      */
6.     public static void main(String[] args) {
7.         // TODO code application logic here
8.         Scanner nhap = new Scanner(System.in);
9.         int n, m, j;
10.            int tao[] = new int [1000];
11.            n = nhap.nextInt(); m = nhap.nextInt(); j =
   nhap.nextInt();
12.            for(int i = 1; i <= j; i++) tao[i] = nhap.nextInt();
13.            int trai = 1, phai = m, sobuoc = 0;
14.            for(int i =1; i <= j; i++){
15.                    if(tao[i] > phai){
16.                            sobuoc += tao[i] - phai;
17.                            trai += tao[i] - phai;
18.                            phai = tao[i];
19.                    }
20.                    else if(tao[i] < trai){
21.                            sobuoc += trai - tao[i];
22.                            phai -= trai - tao[i];
23.                            trai = tao[i];
24.                    }
25.            }
```

```
            System.out.println(sobuoc);
        }

    }
```

5. Dãy con tăng dài nhất

cho một dãy số nguyên gồm N phần tử A[1], A[2], ... A[N].
Biết rằng dãy con tăng đơn điệu là 1 dãy $A[i_1]$,... $A[i_k]$ thỏa mãn
$i_1 < i_2 < ... < i_k$ và $A[i_1] < A[i_2] < .. < A[i_k]$. Hãy cho biết dãy con tăng đơn điệu dài nhất của dãy này có bao nhiêu phần tử?

Input

- Dòng 1 gồm 1 số nguyên là số N ($1 \le N \le 1000$).
- Dòng thứ 2 ghi N số nguyên A[1], A[2], .. A[N] ($1 \le A[i] \le 10000$).

Output

Ghi ra độ dài của dãy con tăng đơn điệu dài nhất.

Ví dụ

Input:

```
6

1 2 5 4 6 2
```

Output:

```
4
```

```
#include<iostream>
using namespace std;
int n,a[1001];
int L[1001];
main(){
   int max = 0;
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   for(int i=1;i<=n;i++){
            L[i] = 1;
            for(int j=1;j<i;j++)
                  if(a[j]<=a[i]&&L[i]<L[j]+1)
                        L[i] = L[j]+1;
       }
       for(int i=1;i<=n;i++)
            if(max<L[i]) max = L[i];
       cout<<max;
  }
```

19.

6. Bãi cỏ gần nhất

Bessie dự định cả ngày sẽ nhai cỏ xuân và ngắm nhìn cảnh xuân trên cánh đồng của nông dân John,

cánh đồng này được chia thành các ô vuông nhỏ với R (1 <= R <= 100) hàng và C (1 <= C <= 100) cột.

Bessie ước gì có thể đếm được số khóm cỏ trên cánh đồng.

Mỗi khóm cỏ trên bản đồ được đánh dấu bằng một ký tự '#' hoặc là 2 ký tự '#' nằm kề nhau (trên đường chéo thì không phải).

Cho bản đồ của cánh đồng, hãy nói cho Bessie biết có bao nhiêu khóm cỏ trên cánh đồng.

Ví dụ như cánh đồng dưới dây với R=5 và C=6:

```
.#....
..#...
..#..#
...##.
.#....
```

Cánh đồng này có 5 khóm cỏ: một khóm ở hàng đầu tiên, một khóm tạo bởi hàng thứ 2 và thứ 3 ở cột thứ 2,

một khóm là 1 ký tự nằm riêng rẽ ở hàng 3, một khóm tạo bởi cột thứ 4 và thứ 5 ở hàng 4, và một khóm cuối cùng ở hàng 5.

Dữ liệu

Dòng 1: 2 số nguyên cách nhau bởi dấu cách: R và C

Dòng 2..R+1: Dòng i+1 mô tả hàng i của cánh đồng với C ký tự, các ký tự là '#' hoặc '.' .

Kết quả

Dòng 1: Một số nguyên cho biết số lượng khóm cỏ trên cánh đồng.

Ví dụ

Dữ liệu

```
5 6
.#....
..#...
..#..#
...##.
.#....
```

Kết quả

```
5
```

```
#include<iostream>

using namespace std;

main(){

   char s[100][100];

   int r, c, dem = 0;

   cin>>r>>c;

   for(int i = 0; i < r; i++){

         int j;

         for(j = 0; j < c; j++) cin>>s[i][j];

         s[i][j] = '\0';

   }

   for(int i = 0; i < r; i++)

         for(int j = 0; j < c; j++)

               if(s[i][j] == '#'){

                     dem ++;

                     s[i][j] = '.';

                     for(int k = i; k < r; k++){
```

```
                        if(s[k][j] == '#') s[k][j] = '.';
                        if(s[k][j] == '.') break;
                    }
                    for(int k = j; k < c; k++){
                        if(s[i][k] == '#') s[i][k] = '.';
                        if(s[i][k] == '.') break;
                    }
                }
    cout<<dem;
}
```

7. Đi xem phim

Nông dân John đang đưa các con bò của anh ta đi xem phim! Xe tải của anh ta thì có sức chứa có hạn thôi, là C (100 <= C <= 5000) kg,

anh ta muốn đưa 1 số con bò đi xem phim sao cho tổng khối lượng của đống bò này là lớn nhất, đồng thời xe tải của anh ta vẫn chịu được.

Cho N (1 <= N <= 16) con bò và khối lượng W_i của từng con, hãy cho biết khối lượng bò lớn nhất mà John có thể đưa đi xem phim là bao nhiêu.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: C và N
- Dòng 2..N+1: Dòng i+1 chứa 1 số nguyên: W_i

Kết quả

- Dòng 1: Một số nguyên là tổng khối lượng bò lớn nhất mà John có thể mang đi xem phim.

Ví dụ:

Dữ liệu:

```
259 5
81
```

```
58

42

33

61
```

Kết quả:

```
242
```

Giải thích: 81+58+42+61 = 242; đây là tổng khối lượng bò lớn nhất có thể được.

```
1. #include<iostream>
2. using namespace std;
3. int n,weight,w[100];
4. int L[100][6000];
5. void Bang(){
6.   for(int i=1;i<=n;i++)
7.         for(int j=1;j<=weight;j++){
8.               if(w[i]<=j)
9.                     L[i][j] = max(L[i-1][j-w[i]]+w[i],L[i-1]
   [j]);
10.                    else L[i][j] = L[i-1][j];
11.              }
12.  }
13.  main(){
14.        cin>>weight>>n;
15.        for(int i=1;i<=n;i++)
16.              cin>>w[i];
17.        Bang();
18.        cout<<L[n][weight];
19.  }
20.
```

8. Đếm hạt

Bessie đã đổ ra bộ sưu tập của cô với N (1<=N<=80) hạt xanh và cam (biểu diễn bởi 0 và 1) xuống sàn.

Bessie dọn dẹp sự lộn xộn bằng cách sắp xếp chúng lại thành 1 đường thẳng dài. Cô muốn đếm số lần hai hạt

liên tiếp khác màu nhau. Hãy giúp cô thực hiện điều này.

INPUT:

* Dòng 1: 1 số nguyên: N

* Dòng 2: Dòng 2 chứa N số nguyên, mỗi số là 0 hoặc 1

OUTPUT:

* Dòng 1: 1 số nguyên biểu diễn số lần hai hạt liên tiếp khác màu.

VÍ DỤ:

INPUT:

```
6
1 0 0 1 1 1
```

SAMPLE OUTPUT :

```
2
```

```
1. #include<iostream>
2. using namespace std;
3. int n,x[100];
4. int nen(){
5.   int max[100];
6.   int kq = 0;
7.   for(int i=1;i<=n;i++){
8.         max[i] = 1;
9.         for(int j=i+1;j<=n;j++)
10.                  if(x[j] == 1-x[j-1]) max[i]++;
11.                  else break;
12.        }
13.        for(int i=1;i<=n;i++)
14.              if(max[i] > kq) kq = max[i];
15.        return kq;
16. }
17. main(){
18.        cin>>n;
19.        for(int i=1;i<=n;i++)
20.              cin>>x[i];
21.        cout<<nen();
22. }
```

9. Ma trận dò mìn

Trong trò chơi dò mìn, người ta cho trước một ma trận cấp n*m trong đó có một số quả mìn ở các vị trí nào đó.

Nhiệm vụ của người chơi là xác định vị trí của các quả mìn này dựa trên các ô xung quanh.

Trong bài này, bạn hãy viết chương trình chuyển từ ma trận n*m mô tả vị trí các quả mìn và

các ô trống thành một ma trận khác trong đó với mỗi ô trống sẽ xác định xem có bao nhiêu quả mìn kề với nó.

Input: Có nhiều bộ test, mỗi bộ test bắt đầu bằng 2 số nguyên n và m (1<=n,m<=100).

Tiếp theo đó là ma trận cấp n*m trong đó vị trí có mìn đánh dấu bởi ký tự * còn không có là dấu chấm (.).

Dòng cuối cùng của file Input chứa hai giá trị 0.

Output: Với mỗi bộ test, in ra màn hình ma trận tương ứng trong đó các vị trí có dấu chấm được thay bằng một số nguyên cho biết số quả mìn kề với ô đó.

```
public class MaTranDoMin {

   static int SoBomKe(String a[], int n, int m, int x, int y) {
      int dem = 0;
      for (int i = x - 1; i < x + 2 && i < n; i++) {
         for (int j = y - 1; j < y + 2 && j < m; j++) {
            if (i >= 0 && j >= 0 && a[i].charAt(j) == '*') {
               dem++;
            }
         }
      }
      if (a[x].charAt(y) == '*') {
         return dem - 1;
      }
      return dem;
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int m, n, i, j;
      String a[];
      do {
         String s = sc.nextLine();
         String st[] = s.split(" ", 2);
         n = Integer.parseInt(st[0]);
         m = Integer.parseInt(st[1]);
         a = new String[n];
         for (i = 0; i < n; i++) {
            a[i] = sc.nextLine();

         }
```

```
            for (i = 0; i < n; i++) {
               for (j = 0; j < m; j++) {
                  if (a[i].charAt(j) != '*') {
                     System.out.print(SoBomKe(a, n, m, i, j));
                  } else {
                     System.out.print(a[i].charAt(j));
                  }
               }
               System.out.println("");
            }
         } while (n != 0 && m != 0);
      }

}
```

10. Ma Trận nghịch đảo

```
public class MaTranNghichDao {


   static void Nhap(double a[][], int n) {

      int i, j;

      Scanner sc = new Scanner(System.in);

      for (i = 0; i < n; i++) {

         for (j = 0; j < n; j++) {

            System.out.print("a[" + i + "][+" + j + "]=");

            a[i][j] = sc.nextDouble();

         }

      }

   }



   static double dinhthuc(double a[][], int n) {

      int i, j, k, sign = 1;

      double temp, det = 1;
```

```
for (i = 0; i < n - 1; i++) {
  if (a[i][i] == 0) // Nếu gặp phần tử trên đường chéo chính bằng 0 thì tìm hàng khác để đổi
  {
    k = i + 1;
    while (k < n && a[k][i] == 0) {
      k++;
    }
    if (k == n) {
      return 0;
    } // Không timg thấy, tức det(a) = 0
    for (j = i; j < n; j++)// Đổi hàng i với hàng k
    {
      temp = a[i][j];
      a[i][j] = a[k][j];
      a[k][j] = temp;
    }
    sign = -sign; // Định thức đổi dấu đó
  }
  for (j = i + 1; j < n; j++)// Biển đổi để các phần tử cùng cột ở hàng dưới bằng 0 đó
  {
    temp = -a[j][i] / a[i][i];
    for (k = i + 1; k < n; k++) {
      a[j][k] += temp * a[i][k]; // Nhân hàng i với (-a[j][i]/a[i][i]) rồi cộng vào hàng j
    }
  }
```

```
        det *= a[i][i]; // Tính dần det(a)
    }

    det *= a[n - 1][n - 1]; // Nhân với phần tử cuối chéo chính nữa là xong
    return det;
}

static void nghichdao(double a[][], double b[][], int n) {
    int i, j, k;
    double temp;
    for (i = 0; i < n; i++) // Tạo ra b là ma trận đơn vị đó
    {
        for (j = 0; j < n; j++) {
            if (i == j) {
                b[i][j] = 1;
            } else {
                b[i][j] = 0;
            }
        }
    }

    for (i = 0; i < n; i++) // Xử lý từ hàng đầu đến hàng cuối
    {
        if (a[i][i] == 0) // Nếu gặp phần tử trên đường chéo chính bằng 0 thì đổi hàng
        {
```

```
        k = i + 1;
        while (k < n && a[k][i] == 0) {
            k++;
        }
        for (j = 0; j < n; j++) // Đổi hàng đó của a đó, cả với ma trận b nữa
        {
            temp = a[i][j];
            a[i][j] = a[k][j];
            a[k][j] = temp;
            temp = b[i][j];
            b[i][j] = b[k][j];
            b[k][j] = temp;
        }
    }

    temp = a[i][i];
    for (j = i; j < n; j++) {
        a[i][j] /= temp;
    }
    for (j = 0; j < n; j++) {
        b[i][j] /= temp;
    }

    for (j = i + 1; j < n; j++) {
        temp = -a[j][i];
```

```
            for (k = i; k < n; k++) {
               a[j][k] += temp * a[i][k];
            }
            for (k = 0; k < n; k++) {
               b[j][k] += temp * b[i][k];
            }
         }
      }// Kết thúc quá trình Gauss

      for (i = n - 1; i > 0; i--) // Bắt đầu quá trình Jordan
      {
         for (j = i - 1; j >= 0; j--) {
            temp = -a[j][i];
            for (k = n - 1; k >= i; k--) {
               a[j][k] += temp * a[i][k];
            }
            for (k = 0; k < n; k++) {
               b[j][k] += temp * b[i][k];
            }
         }
      }

   }

   public static void main(String[] args) {
```

```
    int i, j, n;
    double det;
    double a[][] = new double[10][10];
    double b[][] = new double[10][10];
    // Nhap cap cua ma tran:
    Scanner sc = new Scanner(System.in);
    n = sc.nextInt();
    // Nhap ma tran a:
    Nhap(a, n);
    // Tinh dinh thuc cua a
    det = dinhthuc(a, n);
    //Tinh ma tran nguoc
    if (det != 0) {
       nghichdao(a, b, n);
       for (i = 0; i < n; i++)// In ra ma trận b, bây giờ là ma trận nghich đảo của a
       {
          for (j = 0; j < n; j++) {
             System.out.print(b[i][j] + " ");
          }
          System.out.println("");
       }
    } else {
       System.out.println("\nKhong co ma tran nghich dao\n");
    }
 }
```

11. Ma trận zigzag

```
public class MaTranXigZag {
   /*Ma Tran ZigZag ngang
    EX: 1  2  3  4
       8  7  6  5
       9  10 11 12
       16 15 14 13
    */

   static void ZigZagNgang(int a[][], int n) {
      int dem = 0;
      for (int i = 0; i < n; i++) {
         if (i % 2 == 0) {
            for (int j = 0; j < n; j++) {
               a[i][j] = dem++;
            }
         } else {
            for (int j = n - 1; j >= 0; j--) {
               a[i][j] = dem++;
            }
         }
      }
   }
   /*
    Ma tran ZigZag cheo
```

```
 EX: 1 3 6 10
     2 5 9 13
     4 8 12 15
     7 11 14 16
 */

static void ZigZagCheo(int a[][], int n) {
    int i, s = 1, h = 0, dem = 0;
    while (s <= n * n) {
        for (i = 0; i <= dem; i++) {
            if (h < n) {
                a[h - i][i] = s++;
            } else {
                a[n - 1 - i][h - n + i + 1] = s++;
            }
        }
        h++;
        if (h < n) {
            dem++;
        } else {
            dem--;
        }
    }
}
/*
```

```
ma tran duong cheo zigzag
EX:  1  2  6  7
     3  5  8  13
     4  9  12 14
     10 11 15 16
*/

static void ZigZagDuongCheo(int a[][], int n) {
   int i, s = 1, h = 0, dem = 0;
   while (s <= n * n) {
      if (h % 2 == 0) {
         for (i = 0; i <= dem; i++) {
            if (h < n) {
               a[h - i][i] = s++;
            } else {
               a[n - 1 - i][h - n + i + 1] = s++;
            }
         }
      } else {
         for (i = dem; i >= 0; i--) {
            if (h < n) {
               a[h - i][i] = s++;
            } else {
               a[n - 1 - i][h - n + i + 1] = s++;
            }
```

```
            }
        }
        h++;
        if (h < n) {
            dem++;
        } else {
            dem--;
        }
    }
}

static void show(int a[][], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(a[i][j] + " ");
        }
        System.out.println("");
    }

}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int a[][] = new int[n][n];
```

```
        ZigZagNgang(a, n);
        System.out.println("MT zigzag ngang:");
        show(a, n);
        ZigZagCheo(a, n);
        System.out.println("MT zigzag cheo :");
        show(a, n);
        ZigZagDuongCheo(a, n);
        System.out.println("MT zigzia duong cheo:");
        show(a, n);
    }
}
```

12. Ma trận xoắn ốc

```
public class MaTranXoanOc {


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int sotest = Integer.parseInt(sc.nextLine());
        for (int k = 0; k < sotest; k++) {
            int i, j, n, x, y, row, col;
            int arr[][];


            String s = sc.nextLine();
            String stt[] = s.split(" ");
            n = Integer.parseInt(stt[0]);
            x = Integer.parseInt(stt[1]);
```

```
y = Integer.parseInt(stt[2]);
arr = new int[n][n];
row = n;
col = n;

int value = 1, hangTren = 0, hangDuoi = row - 1, cotTrai = 0, cotPhai = col - 1;
while (value <= row * col) {
   for (i = cotTrai; (i <= cotPhai) && (value <= row * col); i++, value++) {
      arr[hangTren][i] = value;
   }

   hangTren++;

   for (i = hangTren; (i <= hangDuoi) && (value <= row * col); i++, value++) {
      arr[i][cotPhai] = value;
   }
   cotPhai--;

   for (i = cotPhai; (i >= cotTrai) && (value <= row * col); i--, value++) {
      arr[hangDuoi][i] = value;
   }
   hangDuoi--;

   for (i = hangDuoi; (i >= hangTren) && (value <= row * col); i--, value++) {
      arr[i][cotTrai] = value;
```

```
            }

            cotTrai++;

        }

        //Xuat ma tran xoan oc

        System.out.println(arr[x - 1][y - 1]);

    }

  }

}
```

13. Tính toán trên ma trận

Một ma trận N×N được đánh số từ 1 tới N2, theo các đường chéo hình zig-zag.

Con thỏ ở trong ô số 1. Nó có thể nhảy sang các ô kề cạnh (trên, dưới, trái, phải) nếu ô đó tồn tại.

Cho K bước nhảy hợp lệ, viết chương trình tính tổng tất cả các số trên tất cả các ô mà thỏ ghé thăm (tức là mỗi bước đi qua 1 ô thì cộng số ở ô đó vào tổng).

Input

Dòng 1: 2 số nguyên cách nhau bởi dấu cách N và K (1 ≤N ≤ 100 000 , 1 ≤ K ≤ 300 000) lần lượt là kích thước của ma trận và số bước nhảy của con thỏ.

Dòng 2: Chứa dãy có K kí tự có thể là "U","D","L","R" tương ứng với các bước nhảy "lên","xuống","trái","phải". Các bước nhảy đảm bảo rằng sẽ không nhảy ra khỏi ma trận bất cứ lúc nào.

Output

Một số nguyên duy nhất là tổng của các ô mà con thỏ ghé thăm.

```
public class ZigZag_SPOJ {
   static void ZigZagDuongCheo(int a[][], int n) {
      int i, s = 1, h = 0, dem = 0;
      while (s <= n * n) {
         if (h % 2 == 0) {
            for (i = 0; i <= dem; i++) {
               if (h < n) {
                  a[h - i][i] = s++;
               } else {
                  a[n - 1 - i][h - n + i + 1] = s++;
               }
            }
```

```
        } else {
            for (i = dem; i >= 0; i--) {
                if (h < n) {
                    a[h - i][i] = s++;
                } else {
                    a[n - 1 - i][h - n + i + 1] = s++;
                }
            }
        }
        h++;
        if (h < n) {
            dem++;
        } else {
            dem--;
        }
    }
}

public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    String s=sc.nextLine();
    String k=sc.nextLine();
    k=k.trim();
    String st[]=s.split(" ", 2);
    int n=Integer.parseInt(st[0]);
    int sobuoc=Integer.parseInt(st[1]);
    int a[][]=new int[n][n];
    ZigZagDuongCheo(a, n);
    int dem=a[0][0];
    int row=0,col=0;
    for(int i=0;i<sobuoc;i++){

        String t=k.substring(i, i+1);
        switch(t){
            case "U":row--;break;
            case "D":row++;break;
            case "L":col--;break;
            case "R":col++;break;
        }
        dem+=a[row][col];
    }
```

```
        System.out.println(dem);
    }
}
```

III- Sắp Xếp và tìm kiếm

1. Nối bọt

Duyệt nhiều lần từ cuối lên đầu dãy,
tiến hành đổi chỗ 2 phần tử liên tiếp nếu chúng ngược thứ tự. Đến một bước nào đó, khi không có
phép đổi chỗ nào xảy ra thì toàn bộ dãy đã được sắp

```
public class SapXepNoiBot {

    static void init(int a[], int n) {
        Scanner sc = new Scanner(System.in);
        for (int i = 0; i < n; i++) {
            System.out.print("a[" + i + "]=");
            a[i] = sc.nextInt();
        }
    }

    static void result(int a[], int n) {
        for (int i = 0; i < n; i++) {
            System.out.print(a[i] + " ");
        }
    }

    static void bubble_sort(int a[],int n) {
        int i, j, temp;
        for(i=0;i<n-1;i++){
            for(j=0;j<n-1-i;j++){
                if(a[j]>a[j+1]){
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                }
            }
        }
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int a[] = new int[n];
        init(a, n);
        System.out.println("Mang ban dau:");
        result(a, n);
        System.out.println("Mang da sx:");
        bubble_sort(a, n);
        result(a, n);
    }
}
```

2. Sắp xếp chọn

Lựa chọn phần tử có giá trị nhỏ nhất, đổi chỗ cho phần tử đầu tiên. Tiếp theo, lựa chọn phần tử có giá trị nhỏ thứ nhì, đổi chỗ cho phần tử thứ 2. Quá trình tiếp tục cho tới khi toàn bộ dãy được sắp

```
public class SapXepChon {
    static void init(int a[],int n){
        Scanner sc=new Scanner(System.in);
        for(int i=0;i<n;i++){
            a[i]=sc.nextInt();
        }
    }
    static void result(int a[],int n){
        for(int i=0;i<n;i++)System.out.println(a[i]);
    }
    static void selection_sort(int a[],int N) {
        int i, j, k, temp;
        for (i = 0; i < N; i++) {
            k = i;
            for (j = i + 1; j < N; j++) {
                if (a[j] < a[k]) {
                    k = j;
                }
            }
            temp = a[i];
            a[i] = a[k];
            a[k] = temp;
        }
    }
```

```
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        init(a,n);
        selection_sort(a, n);
        result(a, n);
    }
}
```

3. Sắp xếp chèn

Giải thuật này coi như dãy đɑợc chia làm 2 phần. Phần đầu là các phần tử đã được sắp. Từ phần tử tiếp theo, chèn nó vào vị trí thích hợp tại nửa đã sắp sao cho nó vẫn được sắp.
Để chèn phần tử vào nửa đã sắp, chỉ cần dịch chuyển các phần tử lớn hơn nó sang trái 1 vị trí và đưa phần tử này vào vị trí trống trong dãy.

public class SapXepChen {

```
    static void init(int a[],int n){
        Scanner sc=new Scanner(System.in);
        for(int i=0;i<n;i++){
            System.out.print("a["+i+"]=");
            a[i]=sc.nextInt();
        }
    }
    static void result(int a[],int n){
        for(int i=0;i<n;i++)System.out.print(a[i]+" ");
    }
    static void insertion_sort(int a[], int N) {
        int i, j = 0, k, temp = 0;
        for (i = 1; i < N; i++) {
            temp = a[i];
            j = i - 1;
            while ( (j >= 0)&& (a[j] > temp)) {
                a[j + 1] = a[j];
                j--;
            }
            a[j + 1] = temp;
        }

    }
```

```
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        init(a,n);
        System.out.println("Mang ban dau:");
        result(a, n);
        System.out.println("Mang da sx:");
        insertion_sort(a, n);
        result(a, n);
    }
}
```

4. Quick Sort

```
public class QuickSort {
    //Sắp xếp một đoạn từ left-->right
    static void quick(int a[],int n,int left, int right) {
        int i, j;
        int x, y;
        i = left;
        j = right;
        x = a[left];
        do {
            while (a[i] < x && i < right) {
                i++;
            }
            while (a[j] > x && j > left) {
                j--;
            }
            if (i <= j) {
                y = a[i];
                a[i] = a[j];
                a[j] = y;
                i++;
                j--;
            }
        } while (i <= j);
        if (left < j) {
            quick(a,n,left, j);
        }
        if (i < right) {
            quick(a,n,i, right);
```

```
    }
  }
  static void result(int a[],int n){
    for(int i=0;i<n;i++)System.out.print(a[i]+" ");
  }
  public static void main(String[] args) {
    int n=10;
    int a[]={10,6,7,2,19,5,7,21,6,11};
    System.out.println("Day chua sx:");
    result(a, n);
    System.out.println("Day da sx:");
    quick(a, n, 0, n-1);
    result(a, n);
  }
}
```

5. Merge Sort

```
public class MergeSort {
  /* tron 2 mang
   * mang a[l -> m] da sap xep
   * mang a[m+1 -> r] da sap xep
   * */

  static void merge(int []a, int al, int am, int ar){
    int i = al, j = am + 1, k;
    int c[]=new int [ar+1];
    for (k = al; k <= ar; k++) {
      if (i > am) {
        c[k] = a[j++];
        continue;
      }
      if (j > ar) {
        c[k] = a[i++];
        continue;
      }
      if (a[i] < a[j]) {
        c[k] = a[i++];
      } else {
        c[k] = a[j++];
      }
    }
    for (k = al; k <= ar; k++) {
```

```
            a[k] = c[k];
        }
    }

    /* thuat toan sap xep tron*/
    static void merge_sort(int a[], int left, int right){
        int middle;
        if (right <= left) {
            return;
        }
        middle = (right + left) / 2;
        merge_sort(a, left, middle);
        merge_sort(a, middle + 1, right);
        merge(a, left, middle, right);
    }

    static void result(int a[], int n) {
        for (int i = 0; i < n; i++) {
            System.out.print(a[i] + " ");
        }
    }

    public static void main(String[] args) {
        int n = 7;
        int a[] = {2, 6, 3, 8, 5,4,10};
        merge_sort(a, 0, n-1);
        result(a, n);
    }
}
```

6. Heap Sort

```
public class HeapSort {

    private static int[] a;
    private static int n;
    private static int left;
    private static int right;
    private static int largest;

    public static void buildheap(int[] a) {
        n = a.length - 1;
        for (int i = n / 2; i >= 0; i--) {
```

```
        maxheap(a, i);
    }
}

public static void maxheap(int[] a, int i) {
    left = 2 * i;
    right = 2 * i + 1;
    if (left <= n && a[left] > a[i]) {
        largest = left;
    } else {
        largest = i;
    }

    if (right <= n && a[right] > a[largest]) {
        largest = right;
    }
    if (largest != i) {
        exchange(i, largest);
        maxheap(a, largest);
    }
}

public static void exchange(int i, int j) {
    int t = a[i];
    a[i] = a[j];
    a[j] = t;
}

public static void sort(int[] a0) {
    a = a0;
    buildheap(a);

    for (int i = n; i > 0; i--) {
        exchange(0, i);
        n = n - 1;
        maxheap(a, 0);
    }
}

public static void main(String[] args) {
    int[] a1 = {4, 1, 3, 2, 16, 9, 10, 14, 8, 7};
```

```
    sort(a1);
    for (int i = 0; i < a1.length; i++) {
       System.out.print(a1[i] + " ");
    }
 }
```

7. Số bé thứ k

Cho một dãy số nguyên. Bạn hãy tìm số có giá trị thứ k sau khi đã sắp xếp dãy tăng dần.

Input

Dòng đầu chứa 2 số n và k là số phần tử và vị trí cần tìm (1 <= n <= 10^5, 0 <= k < n).

Dòng sau chứa n số nguyên là các phần tử của dãy số.

Output

In ra duy nhất 1 số là đáp án của bài toán.

Example

> **Input:**
>
> 5 3
>
> 5 3 4 8 6
>
> **Output:**

```
8. class Main {
9.
10.          public static void main(String[] args) {
11.              Scanner sc = new Scanner(System.in);
12.              int n = sc.nextInt();
13.              int k = sc.nextInt();
14.              long a[]=new long[n];
15.              for(int i=0;i<n;i++)a[i]=sc.nextLong();
16.              Arrays.sort(a);
17.              System.out.println(a[k]);
18.          }
19.      }
```

7. Sắp xếp 2

Cho một danh sách chứa cả các số và các từ. Yêu cầu bạn hãy sắp xếp danh sách này tăng dần sao cho các từ theo thứ tự từ điển, các số theo thứ tự số. Hơn nữa, nếu phần tử thứ n là số thì danh sách sau khi sắp xếp phần tử thứ n cũng phải là số, nếu là từ thì vẫn là từ.

Lưu ý: Các từ chỉ gồm các chữ in thường trong bảng chữ cái tiếng Anh.

Input

Gồm nhiều dòng, mỗi dòng là một danh sách. Mỗi phần tử của danh sách cách nhau bởi dấu phẩy (“,”) theo sau là dấu cách, và danh sách được kết thúc bằng dấu chấm (“.”).

Dữ liệu kết thúc bởi dòng chỉ chứa một dấu chấm.

Output

Với mỗi danh sách trong dữ liệu, xuất ra danh sách đã sắp xếp thỏa mãn yêu cầu đề bài (có định dạng như trong dữ liệu).

Example

> **Input:**
>
> 0.
>
> banana, strawberry, orange.
>
> banana, strawberry, orange.
>
> 10, 8, 6, 4, 2, 0.
>
> x, 30, -20, z, 1000, 1, y.
>
> 50, 7, kitten, puppy, 2, orangutan, 52, -100, bird, worm, 7, beetle.
>
> .
>
> **Output:**
> 0.
>
> banana, orange, strawberry.
>
> banana, orange, strawberry.
>
> 0, 2, 4, 6, 8, 10.
>
> x, -20, 1, y, 30, 1000, z.
>
> -100, 2, beetle, bird, 7, kitten, 7, 50, orangutan, puppy, 52, worm.

```
1. import java.util.ArrayList;
2. import java.util.Collections;
3. import java.util.Scanner;
4.
5. class Main2 {
6.
7.       public static void main(String[] args) {
8.             Scanner sc = new Scanner(System.in);
9.             while (true) {
10.                         String s = sc.nextLine();
11.                         if (s.equals("."))
12.                               break;
13.                         solve(s);
```

```
14.                 }
15.           }
16.
17.           private static void solve(String s) {
18.                 int n = s.length();
19.                 n--;
20.                 s = s.substring(0, n);
21.                 String a[] = s.split(", ");
22.                 n = a.length;
23.
24.                 boolean position[] = new boolean[n];
25.                 ArrayList<Integer> ai = new ArrayList<>();
26.                 ArrayList<String> as = new ArrayList<>();
27.                 for (int i = 0; i < n; i++) {
28.                       try {
29.                             // System.out.println("i = " + i +
   a[i]);
30.                             int xi = Integer.parseInt(a[i]);
31.                             ai.add(xi);
32.                       } catch (Exception e) {
33.                             as.add(a[i]);
34.                             position[i] = true;
35.                       }
36.                 }
37.
38.                 Collections.sort(ai);
39.                 Collections.sort(as);
40.
41.
42.                 int xi = 0, xs = 0;
43.                 for (int i = 0; i < n - 1; i++) {
44.                       if (!position[i]) {
45.                             System.out.print(ai.get(xi) + ", ");
46.                             xi++;
47.                       } else {
48.                             System.out.print(as.get(xs) + ", ");
49.                             xs++;
50.                       }
51.                 }
52.                 if (position[n - 1])
53.                       System.out.println(as.get(as.size() - 1) +
   ".");
54.                 else
55.                       System.out.println(ai.get(ai.size() - 1) +
   ".");
56.           }
57.     }
```

8. Người trẻ nhất- người già nhất

iết chương trình tìm người trẻ nhất và già nhất trong lớp.

Input

Dòng 1 chứa số n (1<=n<=100), số người trong lớp.

N dòng sau, mỗi dòng là thông tin 1 người có dạng:

personName dd mm yyyy

Trong đó: personName là tên không quá 15 chữ cái, dd,mm,yyyy lần lượt là ngày, tháng, và năm sinh.

Output

Dòng 1: tên người trẻ nhất

Dòng 2: tên người già nhất

Example

> **Input:**
>
> 5
>
> Garfield 20 9 1990
>
> 5
>
> Mickey 1 10 1991
>
> Alice 30 12 1990
>
> Tom 15 8 1993
>
> Jerry 18 9 1990
>
> Garfield 20 9 1990
>
> **Output:**
>
> Tom
>
> Jerry

```
1. #include<iostream>
2. using namespace std;
3. struct hocsinh{
4.       char ten[100];
5.       int ngay, thang, nam;
6. };
7. int tregia(hocsinh a, hocsinh b){
8.                     if(a.nam > b.nam) return -1;
9.                     if(a.nam == b.nam){
10.                                   if(a.thang > b.thang) return
   -1;
11.                                   if(a.thang < b.thang) return
   1;
```

```
12.                                         if(a.thang == b.thang){
13.                                             if(a.ngay > b.ngay) return -1;
14.                                             if(a.ngay < b.ngay) return 1;
15.                                         }
16.                         }
17.                         return 1;
18.     }
19.     main(){
20.             int n, min = 1, max = 1;
21.             hocsinh ds[1001];
22.             cin>>n;
23.             for(int i = 1; i <= n; i++){
24.                 cin>>ds[i].ten;
25.                 cin>>ds[i].ngay>>ds[i].thang>>ds[i].nam;
26.             }
27.             for(int i = 2; i <= n; i++){
28.                 if(tregia(ds[min],ds[i]) == 1) min = i;
29.                 if(tregia(ds[max],ds[i]) == -1) max = i;
30.             }
31.             cout<<ds[min].ten<<"\n"<<ds[max].ten;
32.     }
```

9. Giấy khai sinh

Một buổi họp mặt đại gia đình nhân dịp cụ già Ted tròn 100 tuổi, người ta muốn sắp xếp con cháu của cụ theo thứ tự từ tuổi cao xuống thấp. Giả sử ta có thông tin về giấy khai sinh của từng người đó. Mỗi giấy khai sinh chỉ viết ba thông tin đơn giản gồm: *Tên người cha, Tên người con, Tuổi của người cha lúc sinh con*.

Hãy giúp đại gia đình trên tính ra tuổi của từng người con cháu cụ Ted và viết ra danh sách theo thứ tự từ tuổi cao xuống thấp.

Input

Dòng đầu ghi số bộ test (không quá 100). Với mỗi bộ test:

- Dòng đầu tiên ghi số X (0<X<100) là số người con cháu cần sắp xếp.
- Tiếp theo là X dòng, mỗi dòng ghi thông tin về một giấy khai sinh của từng người (thứ tự ngẫu nhiên) gồm 3 thành phần, mỗi thành phần cách nhau một khoảng trống:
  - Tên người cha: không quá 20 ký tự và không chứa khoảng trống
  - Tên người con: không quá 20 ký tự và không chứa khoảng trống
  - Tuổi của người cha khi sinh con: 1 số nguyên dương, không quá 100.

Output

- Với mỗi bộ test, in ra màn hình thứ tự bộ test (xem thêm trong bộ test ví dụ), sau đó lần lượt là từng người trong danh sách tuổi từ cao xuống thấp (không tính cụ Ted). Mỗi người viết ra hai thông tin: tên, một khoảng trống rồi đến tuổi của người đó.
- Nếu hai người có cùng tuổi thì xếp theo thứ tự từ điển.

Example

Input:

```
2
1
Ted Bill 25
4
Ray James 40
James Beelzebub 17
Ray Mark 75
Ted Ray 20
```

Output:

```
DATASET 1
Bill 75
DATASET 2
Ray 80
James 40
Beelzebub 23
Mark 5
```

```
1.
2. class H {
3.       static String son[];
4.       static String father[];
5.       static int ageHaveSon[];
6.       static int age[];
7.
8.       public static void main(String[] args) {
9.            Scanner sc = new Scanner(System.in);
10.                int testcase = sc.nextInt();
11.
12.                for (int tc = 1; tc <= testcase; tc++) {
13.                     System.out.println("DATASET " + tc);
14.                     int k = sc.nextInt();
15.                     son = new String[k];
16.                     father = new String[k];
17.                     age = new int[k];
18.                     ageHaveSon = new int[k];
19.                     for (int i = 0; i < k; i++) {
20.                          father[i] = sc.next();
21.                          son[i] = sc.next();
22.                          ageHaveSon[i] = sc.nextInt();
23.                     }
24.
25.                     Stack<String> name = new Stack<>();
```

```
                    Stack<Integer> index = new Stack<>();
                    name.push("Ted");
                    index.push(0);
                    while (!name.isEmpty()) {
                        String currentName = name.pop();
                        int currentIndex = index.pop();

                        for (int i = 0; i < k; i++) {
                            if (father[i].equals(currentName)) {
                                index.push(i);
                                name.push(son[i]);

                                if (currentName.equals("Ted")) {
                                    age[i] = 100 - ageHaveSon[i];
                                } else {
                                    age[i] = age[currentIndex] - ageHaveSon[i];
                                }
                            }
                        }

                    }


                    for (int i = 0; i < k; i++) {
                        for (int j = i+1; j < k; j++) {
                            if (age[i] < age[j] ||
                                    (age[i] == age[j] && son[j].compareTo(son[i]) < 0)) {
                                int t = age[i];
                                age[i] = age[j];
                                age[j] = t;

                                String temp = son[i];
                                son[i]=son[j];
                                son[j]=temp;
                            }
                        }
                    }

                    for (int i = 0; i < k; i++) {
                        System.out.println(son[i]+" "+age[i]);
                    }
                }
            }
        }

```

10. Hàng cây

Trên con đường hằng ngày Tí đi học có hàng cây gồm N cây được đánh số từ 1 đến N. Tí muốn biết xem cây nào cao nhất và cây nào thấp nhất, nhưng vì hàng cây dài quá nên Tom không thể xác định được .
Các bạn hãy giúp Tí tìm ra cây cao nhất và cây thấp nhất.

Input

Có nhiều bộ test.

Mỗi bộ test bao gồm: số nguyên dương N (N<= 20) là số cây trong hàng. N dòng tiếp theo, mỗi dòng gồm N số nguyên dương, (mỗi số <= 10^50 và có thể có chữ số 0 ở đầu), là chiều cao của mỗi cây trong hàng.

Input kết thúc bởi số 0.

Output

Với mỗi test, in ra theo mẫu, gồm 2 số nguyên là chiều cao của cây thấp nhất và chều cao của cây cao nhất.

Nếu tất cả các cây có chiều cao bằng nhau, ghi ra “There is a row of trees having equal height.”

Example

```
Input:

5
1
2
3
4
5
3
001
22
333333333333333333333333333333333
3
1
1
1
0


Output:

Case 1: 1 5
Case 2: 1 333333333333333333333333333333333
Case 3: There is a row of trees having equal height.
```

```
1. using namespace std;
```

```
int n;
string a[20];
string chuan(string a){
      while(a[0]=='0'&&a[1]!='\0'){
            a.erase(0,1);
      }
      return a;
}
      int sosanh(string a,string b){
            int A=a.length(),B=b.length(),i=0;
            if(A>B) return 1;
            if(B>A) return -1;
            while(i<A){
                  if(a[i]>b[i]) return 1;
                  if(a[i]<b[i]) return -1;
                  i++;
            }
            return 0;
      }
      void maxmin(int k){
            string min=a[0],max=a[0];
            for(int i=1;i<n;i++){
                  if(sosanh(a[i],max)==1) max=a[i];
                  if(sosanh(a[i],min)==-1) min=a[i];
            }
            if(sosanh(max,min)==0)
                  cout<<"Case "<<k<<": There is a row of trees
  having equal height.\n";
            else
                  cout<<"Case "<<k<<": "<<min<<" "<<max<<"\n";
      }
      main(){
            cin>>n;
            int k=0;
            while(n!=0){
                  k++;
                  for(int i=0;i<n;i++){
                        cin>>a[i];
                        a[i]=chuan(a[i]);
                  }
                  maxmin(k);
                  cin>>n;
            }
      }
```

11. Phân nhóm

Pudge là một anh chàng rất thích hù dọa những người hay đi lẻ trong rừng. Một người được gọi là đi lẻ nếu như chênh lệch chiều cao với những người khác lớn hơn K. Một người

được xếp chung nhóm với nhau nếu như trong nhóm đó, có một người khác có chênh lệch chiều cao với người đó không vượt quá K.

Ví dụ: với tập N = 5 người có các chiều cao: 6, 7, 3, 4, 9 và K = 1 thì ta sẽ có các mối quan hệ sau :

- người thứ 1 và 2 chung một nhóm (chiều cao 6, 7)
- 3 và 4 chung một nhóm (chiều cao 3, 4)
- 5 đi lẻ (chiều cao 9)

Vậy ta sẽ có 3 nhóm: {1, 2}, {3, 4} và {5}

Bạn hãy giúp Pudge tính xem có bao nhiêu nhóm trong rừng để anh còn biết mà hù dọa.

Input

- Dòng đầu chứa 2 số nguyên N, K (0 ≤ N ≤ 10^5, 1 ≤ K ≤ 10^6).
- Dòng thứ hai trong chứa N số nguyên dương – chiều cao của mỗi người (giá trị không vượt quá 10^6).

Output

Dòng duy nhất chứa số nhóm trong rừng.

Example

> **Input:**
>
> 7 1
>
> 2 6 1 7 3 4 9
> **Output:**
>
> 3
>
> Giải thích test;
>
> Nhóm 1 những người có chiều cao: 2 1 3 4
>
> Nhóm 2 gồm: 6 7
>
> Nhóm 3 gồm: 9

```
1. using namespace std;
2. long a[100001], n, k;
3. main(){
4.       cin>>n;
5.       cin>>k;
6.       for(long i = 0; i < n; i++) cin>>a[i];
7.       //sap xep lai mang
8.       sort(a, a + n);
9.       long dem = 1;
10.           for(int i = 0; i < n-1; i++)
11.                if(abs(a[i + 1] - a[i]) > k) dem++;
12.           cout<<dem;
```

```
13.       }
14.
```

12. Chạy đua marathon

1. John cho các con bò của mình chạy đua marathon! Thời gian bò N (1 <= N <= 5,000) về đích được biểu diễn theo dạng Số giờ (0 <= Số giờ <= 99), Số phút (0 <= Số phút <= 59), và số giây (0 <= Số giây <= 59). Để xác định nhà vô địch, John phải sắp xếp các thời gian (theo số giờ, số phút, và số giây) theo thứ tự tăng dần, thời gian ít nhất xếp đầu tiên.

   Ví dụ: Với 3 thời gian như sau:

   11:20:20
   11:15:12
   14:20:14

   Kết quả sau khi sắp xếp là:

   11:15:12
   11:20:20
   14:20:14

   INPUT FORMAT:

   * Line 1: 1 số nguyên: N

   * Lines 2..N+1: Dòng i+1 chứa thời gian bò i được mô tả bởi 3 số nguyên cách bởi dấu cách : Số Giờ , Số Phút, Số giây.

   OUTPUT FORMAT:

   * Dòng 1..N: Mỗi dòng chứa thời gian của 1 con bò là 3 số nguyên cách nhau bởi dấu cách sau khi đã sắp xếp.

   SAMPLE INPUT:

   3
   11 20 20
   11 15 12
   14 20 14

   SAMPLE OUTPUT:

   11 15 12
   11 20 20

```
using namespace std;
2. struct run{
3.       int sec, min, hour;
4.       long time;
5. };
```

```
6. main(){
7.       int n;
8.       run cow[5001];
9.       cin>>n;
10.            for(int i = 0;i < n; i++){
11.                 cin>>cow[i].hour>>cow[i].min>>cow[i].sec;
12.                 cow[i].time = cow[i].hour*3600 + cow[i].min*60 +
   cow[i].sec;
13.           }
14.           for(int i = 0;i < n; i++)
15.                for(int j = 0;j < n; j++)
16.                     if(cow[i].time < cow[j].time)
   swap(cow[i] ,cow[j]);
17.           for(int i = 0;i < n; i++)
18.                cout<<cow[i].hour<<" "<<cow[i].min<<"
   "<<cow[i].sec<<"\n";
19.      }
```

13. Sắp xếp thí sinh

Hãy sắp xếp danh sách thí sinh đã có trong file theo tổng điểm giảm dần.
Mỗi thí sinh gồm các thông tin:
- Mã thí sinh: là một số nguyên, tự động tăng. Tính từ 1.
- Tên thí sinh, ngày sinh
- Điểm môn 1, điểm môn 2, điểm môn 3

Dữ liệu vào: File: F.IN
Dòng đầu chứa số thí sinh. Mỗi thí sinh viết trên 3 dòng:
- Dòng 1: Tên thí sinh
- Dòng 2: Ngày sinh
- Dòng 3,4,5: 3 điểm thi tương ứng. Các điểm thi đều đảm bảo hợp lệ (từ 0 đến 10).

Kết quả: Ghi ra màn hình
In ra danh sách thí sinh đã sắp xếp theo tổng điểm giảm dần. Nếu 2 thí sinh bằng điểm nhau thì thí sinh nào xuất hiện trước trong file sẽ viết trước. Mỗi thí sinh viết trên một dòng gồm: mã, tên, ngày sinh và tổng điểm. Các thông tin cách nhau đúng 1 khoảng trống. Điểm tổng được làm tròn đến 1 số sau dấu phẩy.
F.IN Kết quả
3
Nguyen Van A
12/12/1994
3.5
7.0
5.5
Nguyen Van B
1/9/1994
7.5
9.5

9.5
Nguyen Van C
6/7/1994
4.5
4.5
5.0
Kết quả
2 Nguyen Van B 1/9/1994 26.5
1 Nguyen Van A 12/12/1994 16.0
3 Nguyen Van C 6/7/1994 14.0

Bài làm:

```
public class SXThiSinh {
   static class SinhVien {
      private int id;
      private String ten;
      private String ngaySinh;
      private float diem1;
      private float diem2;
      private float diem3;
      get/set();
   public static void main(String[] args) throws FileNotFoundException {
      Scanner sc = new Scanner(new File("ThiSinh.txt"));
      int n = Integer.parseInt(sc.nextLine());
      ArrayList<SinhVien> list = new ArrayList<>();
      for (int i = 0; i < n; i++) {
         SinhVien sv = new SinhVien();
         sv.setId(i+1);
         sv.setTen(sc.nextLine());
         sv.setNgaySinh(sc.nextLine());
         sv.setDiem1(Float.parseFloat(sc.nextLine()));
         sv.setDiem2(Float.parseFloat(sc.nextLine()));
         sv.setDiem3(Float.parseFloat(sc.nextLine()));
         list.add(sv); }
      for (int i = 0; i < list.size(); i++) {
         for (int j = 0; j < list.size(); j++) {
            if (list.get(i).getTongDiem() > list.get(j).getTongDiem()) {
               SinhVien temp = new SinhVien();
               temp = list.get(i);
               list.set(i, list.get(j));
               list.set(j, temp);}}}
      for (int i = 0; i < list.size(); i++) {
         System.out.println(list.get(i).id + " "+list.get(i).ten + " "+list.get(i).ngaySinh + "
"+list.get(i).getTongDiem());}}}
```

14. Số bé thứ k

Cho một dãy số nguyên. Bạn hãy tìm số có giá trị thứ k sau khi đã sắp xếp dãy tăng dần.

Input

Dòng đầu chứa 2 số n và k là số phần tử và vị trí cần tìm (1 <= n <= 10^5, 0 <= k < n).

Dòng sau chứa n số nguyên là các phần tử của dãy số.

Output

In ra duy nhất 1 số là đáp án của bài toán.

Example

> **Input:**
>
> 5 3
>
> 5 3 4 8 6
>
> **Output:**
> 6

```
1. class Main {
2.
3.     public static void main(String[] args) {
4.         Scanner sc = new Scanner(System.in);
5.         int n = sc.nextInt();
6.         int k = sc.nextInt();
7.         long a[]=new long[n];
8.         for(int i=0;i<n;i++)a[i]=sc.nextLong();
9.         Arrays.sort(a);
10.              System.out.println(a[k]);
11.          }
12.     }
13.
```

V- Cấu trúc dữ liệu ngăn xếp

1. Xổ số

> Công ty điện thoại LuckyPhone tổ chức xổ số để quyên góp tiền cho quỹ hỗ trợ tài năng trẻ nước NumberLand. Luật chơi như sau:
>
> - Có n khách hàng tham gia chơi ($1 \le n \le 10^6$) mỗi khách hàng tự chọn một số tự nhiên có không quá 9 chữ số, sau đó soạn tin nhắn gửi số mình chọn đến số điện thoại 19001234.

Mỗi khách hàng chỉ được gửi đúng một tin nhắn.

- Trong các số nhận được từ các khách hàng, số lớn nhất trong các số được ít khách hàng gửi đến nhất sẽ được chọn làm số may mắn (Lucky Number).

- Công ty thu từ mỗi khách hàng đã gửi tin nhắn x đồng ($1 \le x \le 10^9$) và phải tặng cho tất cả các khách hàng đã gửi số may mắn mỗi người một giải thưởng giá trị đúng bằng số may mắn.

- Lợi nhuận của công ty thu được trong cuộc chơi bằng tổng số tiền thu được từ các khách hàng trừ đi số tiền trao giải thưởng.

Yêu cầu: Hãy tính lợi nhuận của công ty LuckyPhone trong cuộc chơi

Dữ liệu:

- Dòng 1: Chứa hai số nguyên dương n,x

- n dòng tiếp theo, mỗi dòng chứa một số nhận được từ một khách hàng

Kết quả:

- Ghi ra lợi nhuận thu được.

Các số trên một dòng của Input/Output files được/phải ghi cách nhau ít nhất một dấu cách

Ví dụ:

INPUT	OUTPUT
9 100 2 33 1 2 444 33 1 1 444	12

```
1. class MainCore {
2.     public static void main(String[] args) throws Exception {
3.         Scanner sc = new Scanner(System.in);
4.         long n = sc.nextLong(), x = sc.nextLong();
5.         HashMap<Long, Long> hm = new HashMap<>();
6.         for (long i = 0; i < n; i++) {
```

```
            long k = sc.nextLong();
            if (!hm.containsKey(k)) hm.put(k, 1L);
            else hm.put(k, hm.get(k) + 1);
        }
        long maxFre = 0, maxNum = -1;
        for (Long key : hm.keySet())
            if (hm.get(key) >= maxFre && key > maxNum) {
                maxNum = key;
                maxFre = hm.get(key);
            }
        System.out.println(n * x - maxFre * maxNum);
        sc.close();
    }
}
```

2. Số gần may mắn

Petya yêu thích các số may mắn. Ta biết rằng một số là *số may mắn* nếu biểu diễn thập phân của nó chỉ chứa các chữ số may mắn là 4 và 7.

Ví dụ, các số 47, 744, 4 là số may mắn và 5, 17, 467 không phải.

Tuy nhiên, không phải tất cả các số là may mắn. Petya gọi 1 số là số *gần may mắn* nếu có số chữ số may mắn là số may mắn.

Hãy kiểm tra xem 1 số có là số gần may mắn không.

Input

Chỉ chứ số nguyên n ($1 \le n \le 10^{18}$).

Output

In ra trên 1 dòng "YES" nếu *n* là *số gần may mắn*. Ngược lại, in ra "NO" (không có dấu nháy).

Ví dụ:

Input

40047

Output

NO

Input

7747774

Output

YES

Input

1000000000000000000

Output

NO

Giải thích:

Trong test đầu, có 3 chữ số may mắn (1 ở đầu và 2 ở cuối), mà 3 không là số may mắn, nên đáp án là "NO".

Trong test thứ 2, có 7 chữ số may mắn, mà 7 là số may mắn, nên đáp án là "YES".

Trong test thứ 3, không có chữ số may mắn nào, nên đáp án là "NO

```
1. using namespace std;
2. void ganmm(long long n){
3.       int so = 0;
4.       while(n>0){
5.             int i = n%10;
6.             if(i == 4 || i == 7) so++;
7.             n /= 10;
8.       }
9.       if(so == 4 || so == 7) cout<<"YES";
10.            else cout<<"NO";
11.      }
12.      main(){
13.            long long n;
14.            cin>>n;
15.            ganmm(n);
16.      }
```

3. Lại số gần may mắn

Petya yêu thích các số may mắn. Ta biết rằng một số là *số may mắn* nếu biểu diễn thập phân của nó chỉ chứa các chữ số may mắn là 4 và 7.

Ví dụ, các số 47, 744, 4 là số may mắn và 5, 17, 467 không phải.

Petya muốn tìm số may mắn bé nhất có tổng các chữ số bằng n. Hãy giúp anh ấy.

Input

Một dòng chứa số nguyên n ($1 \le n \le 10^6$) — tổng các chữ số của số may mắn cần tìm.

Output

In ra trên 1 dòng số may mắn bé nhất, mà tổng các chữ số bằng n. Nếu không tồn tại số thỏa mãn, in ra -1.

Ví dụ:

Input

11

> Output
> 47
>
> Input
> 10
> Output
> -1

```
class CF {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt(), iMin = 0, jMin = 0;

        for (int i = n / 7; i >= 0; i--) {

            int j = (n - i * 7) / 4;

            // System.out.println("i=" + i + " - j="+j);

            if (i * 7 + j * 4 == n) {

                iMin = i;

                jMin = j;

                break;

            }

        }

        if (iMin == 0 && jMin == 0) {

            System.out.println(-1);

            return;

        }

        for (int i = 0; i < jMin; i++) System.out.print(4);
```

```
        String s100 =
"77777777777777777777777777777777777777777777777777777777777777777777777777777777
77777777777777777777777777777777";

        while (iMin >= 100) {

            System.out.print(s100);

            iMin -= 100;

        }

        for (int i = 0; i < iMin; i++) System.out.print(7);

        System.out.println();

    }

}
```

4. Nguyên tố hóa học

PTIT121E - Nguyên tố hóa học

Hóa chất chỉ gồm các nguyên tố C, H, O có trọng lượng 12,1, 16 tương ứng.

Nó được biểu diễn dạng "nén", ví dụ COOHHH là CO2H3 hay CH (CO2H) (CO2H) (CO2H) là CH(CO2H)3. Nếu ở dạng nén thì số lần lặp >=2 và <=9.

Tính khối lượng hóa chất.

Input

Gồm một dòng mô tả hóa chất không quá 100 kí tự chỉ gồm C, H, O, (,), 2,..,9.

Output

Khối lượng của hóa chất (luôn <=10000).

Example

INPUT	OUTPUT
COOH	45

INPUT	OUTPU

	T
CH(CO2H)3	148

INPUT	OUTPUT
((CH)2(OH2H)(C(H))O)3	222

```
1. class CF {
2.
3.     public static void main(String[] args) {
4.         Scanner sc = new Scanner(System.in);
5.         String s = sc.next();
6.         Stack<Integer> st = new Stack<Integer>();
7.         for (int i = 0; i < s.length(); i++) {
8.             char c = s.charAt(i);
9.             switch (c) {
10.                     case '(':
11.                         st.push(0);
12.                         break;
13.                     case 'C':
14.                         st.push(12);
15.                         break;
16.                     case 'H':
17.                         st.push(1);
18.                         break;
19.                     case 'O':
20.                         st.push(16);
21.                         break;
22.                     case ')':
23.                         int d = 0;
24.                         while (st.peek() != 0) {
25.                             d += st.pop();
26.                         }
27.                         st.pop();st.push(d);
28.                         break;
29.                 }
30.                 if(c>='2'&&c<='9'){
31.                     int u=st.pop()*(c-'0');
32.                     st.push(u);
33.                 }
34.             }
35.             int sum=0;
36.             while(!st.isEmpty()) sum+=st.pop();
37.             System.out.println(sum);
38.     }
```

5. Ngăn xếp cơ bản

Thao tác:

- 1. ‘init’ : Khởi tạo stack rỗng.
- 2. ‘push x’: Thêm phần tử x vào stack. (x là số nguyên dương không quá 10 mũ 9)
- 3. ‘pop’: Nếu stack không rỗng lấy ra phần tử ở đỉnh stack.
- 4. ‘top’: Trả về phần tử ở đỉnh stack. Nếu stack rỗng, trả về -1.
- 5. ‘size:’ Trả về kích thước stack (số phần tử hiện tại của stack).
- 6. ‘empty’: Kiểm tra stack rỗng hay không, nếu rỗng trả về 1, ngược lại là 0.
- 7. ‘end’: Kết thúc chương trình.

Dữ liệu:

Gồm nhiều dòng mô tả các thao tác như trên (số phần tử của stack luôn không quá 1000).

Kết quả:

Khi gặp các thao tác 4,5,6 các bạn in ra trên 1 dòng tương ứng với câu trả lời.

```
public class BasicStack {

  static class _Stack {

    private int size;
    private int a[]= new int[1000];;

    public void Init() {
      size = 0;
    }

    public void Push(int x) {
      a[size] = x;
      size++;
    }

    public int Pop(){
```

```
        if (this.empty() == 0) {
            size--;
            return a[size];
        }
        return 0;
    }

    public int Top() {
        if (this.empty() == 0) {
            return a[size - 1];
        } else {
            return -1;
        }
    }

    public int Size() {
        return size;
    }

    public int empty() {
        if (size == 0) {
            return 1;
        } else {
            return 0;
        }
    }

}

public static void main(String[] args){
    _Stack st = new _Stack();
    Scanner sc = new Scanner(System.in);
    String action = sc.nextLine();
    action=action.trim();
    while (!action.equals("end")) {
        switch (action) {
            case "init":
                st.Init();
                break;
            case "pop":
                if(st.empty()==0)   st.Pop();
```

```
                break;
            case "top":
                System.out.println(st.Top());
                break;
            case "empty":
                System.out.println(st.empty());
                break;
            case "size":
                System.out.println(st.Size());
                break;
        }
        if(action.startsWith("push")){
            int x=Integer.parseInt(action.substring(5));
            st.Push(x);
        }
        action=sc.nextLine();
        action=action.trim();
    }
  }
}
```

6. Chuyển đổi cơ số

```
public class ChuyenDoiCoSo {

  //tu thap phan sang cac he co so khac
  static String ThapPhan(String so, int coso) {
    long n = Long.parseLong(so);
    Stack st = new Stack();
    while (n != 0) {
      long k = n % coso;
      n /= coso;
      st.push(k);
    }
    String s = "";
    while (!st.empty()) {
      long t = (long) st.pop();
      if (t >= 10) {
        s += (char) ('A' + (t - 10));
      } else {
        s += t;
      }
```

```
    }
    return s;
}

//Chuyen tu co so khac sang thap phan
static long toThapPhan(String so, int coso) {
    int d = so.length();
    long dec = 0;
    for (int i = 0; i < so.length(); i++) {
        long k = so.charAt(i);
        if (k <= 57) {
            dec += (k - 48) * Math.pow(coso, d - 1);

        } else {
            dec += (k - 65 + 10) * Math.pow(coso, d - 1);
        }
        d--;
    }
    return dec;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = Integer.parseInt(sc.nextLine());
    for (int i = 0; i < n; i++) {
        String line = sc.nextLine();
        String st[] = line.split(" ");
        int hecoso1 = Integer.parseInt(st[0]);
        int hecoso2 = Integer.parseInt(st[1]);
        String so = st[2];
        if (hecoso1 == 10) {
            System.out.println(hecoso1 + " " + so);
            System.out.println(hecoso2 + " " + ThapPhan(so, hecoso2));
            System.out.println("");
        } else if (hecoso2 == 10) {
            System.out.println(hecoso1 + " " + so);
            System.out.println(hecoso2 + " " + toThapPhan(so, hecoso1));
            System.out.println("");
        } else {
            System.out.println(hecoso1 + " " + so);
            long k = toThapPhan(so, hecoso1);
```

```
            System.out.println(hecoso2 + " " + ThapPhan("" + k, hecoso2));
            System.out.println("");
        }
      }

   }
}
```

7. Đảo xâu

Một chuỗi được gọi là đối xứng (palindrome) nếu như khi đọc chuỗi này từ phải sang trái cũng thu được chuỗi ban đầu.

Yêu cầu: tìm một chuỗi con đối xứng dài nhất của một chuỗi s cho trước. Chuỗi con là chuỗi thu được khi xóa đi một số ký tự từ chuỗi ban đầu.

Dữ liệu vào
Gồm một dòng duy nhất chứa chuỗi s, chỉ gồm những chữ cái in thường.

Kết qủa
Gồm một dòng duy nhất là một xâu con đối xứng dài nhất của xâu s. Nếu có nhiều kết quả, chỉ cần in ra một kết quả bất kỳ.

Giới hạn
Chuỗi s có độ dài không vượt quá 2000

```
public class DaoXau {
   static String dao(String s){
      Stack st=new Stack();
      String s2="";
      for(int i=0;i<s.length();i++)st.push(s.substring(i, i+1));
      while(!st.empty()){
         s2+=st.pop();
      }
      return s2;
   }
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      String s=sc.nextLine();
      System.out.println(dao(s));
   }
```

8. Dấu ngặc đúng

Cho các đoạn văn chứa các dấu ngoặc, có thể là ngoặc đơn đơn (“()”) hoặc ngoặc vuông (“[]”).

Một đoạn văn đúng là đoạn mà với mỗi dấu mở ngoặc thì sẽ có dấu đóng ngoặc tương ứng và đúng thứ tự.

Nhiệm vụ của bạn kiểm tra xem đoạn văn có đúng hay không.

Input

Gồm nhiều bộ test, mỗi bộ test trên một dòng chứa đoạn văn cần kiểm tra có thể bao gồm:

các kí tự trong bảng chữ cái tiếng Anh, dấu cách, và dấu ngoặc (ngoặc đơn hoặc ngoặc vuông).

Kết thúc mỗi bộ test là một dấu chấm. Mỗi dòng có không quá 100 kí tự.

Dữ liệu kết thúc bởi dòng chứa duy nhất một dấu chấm.

Output

Với mỗi bộ test, xuất ra trên một dòng “yes” nếu đoạn văn đúng, ngược lại in ra “no”.

```
public class DauNgoacDung {


   public static void main(String[] args) throws FileNotFoundException {
      Stack st = new Stack();
      Scanner sc = new Scanner(System.in);
      String input = sc.nextLine();
      while (!input.equals(".")) {
         boolean kt = true;
         st.clear();
         for (int i = 0; i < input.length(); i++) {
            char t = input.charAt(i);
            if (t == '(' || t == '[') {
```

```
                st.push(t);
            }
            if (t == ')' || t == ']') {
                if (st.empty() || (t == ']' && (char) st.peek() == '(') || (t == ')' && (char) st.peek() == '[')) {
                    kt = false;
                    break;
                } else {
                    st.pop();
                }
            }
        }
        if (st.empty() && kt) {
            System.out.println("yes");
        }else{
            System.out.println("no");
        }
        input = sc.nextLine();

    }
  }
}
```

9. Tính biểu thức hậu tố

```
public class TinhToanBieuThucHauTo {

  static int pri(char x) {
```

```
    if (x == '+' || x == '-') {
        return 1;
    }
    if (x == '*' || x == '/' || x == '%' || x == '^') {
        return 2;
    }
    return 0;
}

static String HauTo(String s) {

    String EH = "";
    Stack st = new Stack();
    s = s.trim();
    s = s.replaceAll(" ", "");
    for (int i = 0; i < s.length(); i++) {
        char t = s.charAt(i);
        if (t >= '0' && t <= '9') {
            if ((i + 1) < s.length() && (s.charAt(i + 1) >= '0' && s.charAt(i + 1) <= '9')) {
                EH += t;
            } else {
                EH += t + " ";
            }
        } else if (t == '(') {
            st.push(t);
```

```
        } else if (t == ')') {
            char x = (char) st.pop();
            while (x != '(') {
                EH += x + " ";
                x = (char) st.pop();
            }
        } else {
            while (!st.empty() && pri((char) st.peek()) >= pri(t)) {
                EH += st.pop() + " ";
            }
            st.push(t);
        }
    }
    while (!st.empty()) {
        EH += st.pop() + " ";
    }
    System.out.println("HT:" + EH);
    return EH;
}

static int Tinh(String s) {
    String post[] = s.split(" ");
    Stack st = new Stack();
    for (String post1 : post) {
        int top;
```

```
switch (post1) {
    case "+":
        st.push((int) st.pop() + (int) st.pop());
        break;
    case "-":
        top = (int) st.pop();
        st.push((int) st.pop() - top);
        break;
    case "*":
        st.push((int) st.pop() * (int) st.pop());
        break;
    case "/":
        top = (int) st.pop();
        st.push((int) st.pop() / top);
        break;
    case "%":
        top = (int) st.pop();
        st.push((int) st.pop() % top);
        break;
    case "^":
        top = (int) st.pop();
        st.push(Math.pow((int) st.pop(), top));
        break;
    default:
        st.push(Integer.parseInt(post1));
```

```
        }
      }
      return (int) st.peek();
    }

    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String In = sc.nextLine();
      System.out.println(Tinh(HauTo(In)));
    }
}
```

10. Tính biểu thức trung tố

```
public class TinhToanBieuThucTrungTo {

  static int pri(String x) {
    if ("+".equals(x) || "-".equals(x)) {
      return 1;
    }
    if ("*".equals(x) || "/".equals(x) || "%".equals(x) || "^".equals(x)) {
      return 2;
    }
    if (x.equals("(") || x.equals(")")) {
      return 0;
    }
    return -1;
  }

  static int tt(int a, int b, String toantu) {
    switch (toantu) {
      case "+":
        return b + a;
      case "-":
        return b - a;
      case "*":
```

```
            return b * a;
        case "/":
            return b / a;
        case "^":
            return (int) Math.pow(b, a);
        case "%":
            return b % a;
    }
    return 0;
}
/*
 bieu thuc  duoc chuan hoa giua cac thanh phan co ngan canh boi " "
 */
static String Format(String s) {
    s = s.replaceAll(" ", "");
    String st = "";
    for (int i = 0; i < s.length(); i++) {
        st = st + (s.substring(i, i + 1) + " ");
    }
    return st.trim();
}

static int Tinh(String s) {
    Stack sh = new Stack();
    Stack st = new Stack();
    s = Format(s);
    String post[] = s.split(" ");

    for (String post1 : post) {
        if (pri(post1) == -1) {
            sh.push(Integer.parseInt(post1));
        }
        if (post1.equals("(")) {
            st.push(post1);
        }
        if (pri(post1) > 0) {
            while (!st.empty() && pri(post1) <= pri((String) st.peek())) {
                sh.push(tt((int) sh.pop(), (int) sh.pop(), (String) st.pop()));
            }
            st.push(post1);
        }
```

```
            if (post1.equals(")")) {
                while (!st.peek().equals("(")) {
                    sh.push(tt((int) sh.pop(), (int) sh.pop(), (String) st.pop()));
                }
                st.pop();
            }
        }
        while (!st.empty()) {
            sh.push(tt((int) sh.pop(), (int) sh.pop(), (String) st.pop()));
        }
        return (int) sh.peek();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String In = sc.nextLine();
        System.out.println(Tinh(In));
    }
}
```

VI – Thuật toán tham lam

1. Đoạn số có tổng bằng nhau

Một đoạn số có tổng bằng nhau trong một dãy số là một nhóm các số theo đúng thứ tự ban đầu trong dãy mà nếu nhóm với nhau thì sẽ cho ra cùng một giá trị tổng. Ví dụ với dãy: 2 5 1 3 3 7 thì ta có thể nhóm thành: (2 5) (1 3 3) (7) cùng cho giá trị tổng là 7.

Chú ý: đoạn đặc biệt chứa tất cả các phần tử của dãy cũng được coi là một đoạn có tổng bằng nhau với chính giá trị tổng các số của dãy đó.

Yêu cầu: viết chương trình nhận vào các dãy số nguyên dương và trả về giá trị ***tổng nhỏ nhất*** có thể của một đoạn tổng bằng nhau trong dãy.

Dữ liệu vào

Dòng đầu tiên chứa một số nguyên 1 ≤ t ≤ 1000 là số lượng bộ test. Mỗi bộ test bao gồm:

- Dòng đầu tiên chứa thứ tự bộ test và số M (1≤ M ≤ 10000) là số phần tử của dãy.
- Các dòng tiếp theo mỗi dòng ghi 10 số của dãy phân cách bởi 1 dấu cách. Dòng cuối cùng có thể có ít hơn 10 số. (Các số trong dãy đều nhỏ hơn 20000).

Dữ liệu ra

Với mỗi bộ test, in ra trên một dòng gồm số thứ tự bộ test và tổng nhỏ nhất có thể đạt được của các đoạn số có tổng bằng nhau.

Example

INPUT	OUTPUT	
3 1 6 2 5 1 3 3 7 2 6 1 2 3 4 5 6 3 20 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1 2 1 1		

```
1. class Main {
2.     static Scanner sc = new Scanner(System.in);
3.     static int t, n, a[];
4.
5.     static void solve() {
6.         t = sc.nextInt();
7.         n = sc.nextInt();
8.         a = new int[n + 1];
9.         a[0] = 0;
10.             for (int i = 1; i <= n; i++) a[i] = a[i - 1] +
   sc.nextInt();
11.             int s=a[n];
12.             for (int i = 1; i < n; i++)
13.                 if (s % a[i] == 0) {
14.                     int k = 2;
15.                     for (int j = i + 1; j < n; j++) {
16.                         if (a[j] > a[i] * k) break;
17.                         else if (a[j] == a[i] * k) k++;
18.                     }
19.                     if (a[i] * k == s) {
20.                         System.out.println(t+" "+a[i]);
21.                         return;
22.                     }
23.                 }
24.             System.out.println(t+" "+s);
25.         }
26.
27.
28.         public static void main(String[] args) {
29.             int test = sc.nextInt();
30.             while (test-- > 0) {
31.                 solve();
32.             }
33.
```

2. Hình lập phương

Hãy tưởng tượng bạn đang chơi một trò chơi máy tính đơn giản. Có n hình lập phương được tô bởi m màu. Bạn được cho phép xóa bỏ không quá k hình lập phương (không nhất thiết phải là k hình lập phương liên tiếp). Sau khi xóa bỏ, dồn tất cả các hình lập phương lại với nhau để không còn các khoảng trống và bắt đầu tính điểm. Điểm số của bạn sẽ bằng độ dài lớn nhất của chuỗi các hình lập phương liên tiếp cùng màu. Hãy tính xem, số điểm lớn nhất bạn có thể có được là bao nhiêu.

Input

Dòng đầu tiên chứa 3 số nguyên n, m, k (1 <= n <= 2.10^5, 1 <= m <= 10^5, 0 <= k < n).

Dòng thứ 2 chứa n số nguyên có giá trị từ 1 đến m, đại diện cho màu của n hình lập phương.

Output

Dòng duy nhất là số điểm lớn nhất bạn có được.

Example

Test 1:

Input:

10 3 2

1 2 1 1 3 2 1 1 2 2

Output:

4

3.

```
4. lass ABC {
5.     public static void main(String[] args) {
6.         Scanner sc = new Scanner(System.in);
7.         int n = sc.nextInt(), m = sc.nextInt(), k = sc.nextInt();
8.         ArrayList<ArrayList<Integer>> arr = new
   ArrayList<ArrayList<Integer>>();
9.         for (int i = 0; i < m; i++) {
10.                 arr.add(new ArrayList<Integer>());
11.             }
12.
13.             for (int i = 0; i < n; i++) {
14.                 int t = sc.nextInt() - 1;
15.                 arr.get(t).add(i);
16.             }
17.
18.             int MAX = 0;
```

```
            for (int i = 0; i < m; i++) {
                int t = 0, max;
                for (int j = 1; j < arr.get(i).size(); j++) {
                    max = 0;
                    t = 0;
                    int pos = j;
                    while (t <= k && pos < arr.get(i).size()) {

                        t += arr.get(i).get(pos) -
 arr.get(i).get(pos - 1) - 1;
                        if (t > k) break;
                        max++;
                        pos++;

                    }
                    if (max > MAX) {
                        MAX = max;
    //                      System.out.println(pos);
    //                      System.out.println(j);
                    }
                }
            }

            System.out.println(MAX + 1);
        }
    }
```

3. Số siêu tự nhiện

Một số siêu tự nhiên là một chuỗi chứa các chữ số và dấu hỏi (ví dụ như 36?1?8). Một số X phù hợp với một số siêu tự nhiên W nếu X có thể được tạo thành từ W bằng cách thay thế dấu hỏi bằng các chữ số tùy ý. Ví dụ: 365198 phù hợp với số siêu tự nhiên 36?1?8, nhưng 360199, 361028 hoặ 36128 thì không. Viết chương trình đọc số siêu tự nhiên W và số X (có cùng độ dài n), và xác định số các số phù hợp với W và lớn hơn X.

Dữ liệu:

Gồm nhiều bộ test, mỗi bộ test gồm 2 dòng:

- Dòng 1 chứa số siêu tự nhiên W
- Dòng 2 chứa số nguyên X

Độ dài các số từ 1 đến 10 kí tự.

Dữ liệu kết thúc bởi dấu #

Kết quả:

Với mỗi bộ test, in ra trên 1 dòng số các số phù hợp với W và lớn hơn X.

Example

```
Input:
36?1?8
236428
8?3
910
?
5
#

Output:
100
0
4
```

4. Số đối xứng 2

5. Tìm dãy số

Cho trước một dãy số dương có N phần tử. Bạn biết trước tổng của bất kì 2 phần tử nào trong dãy số, hãy tìm dãy số ban đầu.

Input

Dòng đầu tiên là N, số phần tử của dãy số. (2 <= N <= 1000)

N dòng sau, mỗi dòng gồm N số (mỗi số <= 100 000) mô tả ma trận biểu diễn tổng của 2 phần tử trong dãy.

* S(i,j) = 0 nếu i = j.

* S(i,j) = A[i] + A[j] với i ≠ j, là tổng của phần tử thứ i và thứ j trong dãy số.

Output

In ra trên 1 dòng dãy số cần tìm. Input luôn đảm bảo có 1 đáp số duy nhất.

Example

Input1:

2
0 2
2 0

Ouput1:

1 1

Input2:

4

0	3	6	7
3	0	5	6
6	5	0	9

7 6 9 0

Ouput2:

2 1 4 5

```
1. class Main2 {
2.       public static void main(String[] args) {
3.             Scanner sc = new Scanner(System.in);
4.             int n = sc.nextInt(), a[][] = new int[n][n];
5.             for (int i = 0; i < n; i++)
6.                   for (int j = 0; j < n; j++)
7.                         a[i][j] = sc.nextInt();
8.             long s = 0;
9.             for (int i = 0; i < n; i++)
10.                        s += (long) a[0][i];
11.                  long t = 0;
12.                  for (int i = 1; i < n; i++)
13.                        for (int j = i + 1; j < n; j++)
14.                              t += (long) a[i][j];
15.                  long b[]=new long [n];
16.                  b[0]= (s-t/(n-2))/(n-1);
17.                  for (int i = 1; i < n; i++) b[i]=a[0][i]-b[0];
18.                  for (int i = 0; i < n; i++)System.out.print(b[i]
   +" ");
19.                  System.out.println();
20.            }
21.
22.      }
23.
```

21. Biểu thức

Một dãy gồm n số nguyên không âm a_1, a_2,..., a_n được viết thành một hàng ngang, giữa hai số liên tiếp có một khoảng trắng, như vậy có tất cả (n-1) khoảng trắng. Người ta muốn đặt k dấu cộng và (n-1-k) dấu trừ vào (n-1) khoảng trắng đó để nhận được một biểu thức có giá trị lớn nhất.

Ví dụ, với dãy gồm 5 số nguyên 28, 9, 5, 1, 69 và k = 2 thì cách đặt 28+9-5-1+69 là biểu thức có giá trị lớn nhất.

Yêu cầu: Cho dãy gồm n số nguyên không âm a1, a2,..., an và số nguyên dương k, hãy tìm cách đặt k dấu cộng và (n-1-k) dấu trừ vào (n-1) khoảng trắng để nhận được một biểu thức có giá trị lớn nhất.

Input

- Dòng đầu chứa hai số nguyên dương n, k ($k < n \leq 10^5$);
- Dòng thứ hai chứa n số nguyên không âm a_1, a_2,..., a_n ($a_n \leq 10^6$)

Output

- Một số nguyên là giá trị của biểu thức đạt được.

Example

```
Input:

5 2

28 9 5 1 69

Output:
100
```

```
1. class Main2 {
2.       public static void main(String[] args) {
3.             Scanner sc = new Scanner(System.in);
4.             int n = sc.nextInt(), k = sc.nextInt();
5.             long b=sc.nextLong();
6.             int a[]=new int[n-1];
7.             for(int i=0;i<n-1;i++)a[i]=sc.nextInt();
8.             Arrays.sort(a);
9.             for(int i=0;i<n-1-k;i++) b-=(long)a[i];
10.                  for(int i=n-1-k;i<n-1;i++) b+=(long)a[i];
11.                  System.out.println(b);
12.            }
13.
14.
15.      }
```

22. Biểu thức
23. Sd
24. Ds

sd

4. Ruy Băng

Hôm nay Ryze tham gia một BigGame, trong đó có trò cắt ruy bang. Ryze phải cắt ruy bang có chiều dài n thành các mảnh thỏa mãn:

• Các mảnh có chiều dài là a hoặc b hoặc c

• Số mảnh cắt được là nhiều nhất

Hãy giúp Ryze xác định số mảnh nhiều nhất có thể cắt được.

Input

Một dòng chứa 4 số nguyên n, a, b, và c (1 <= n, a, b, c <= 4000)

Output

Đáp án của bài toán. Đề bài đảm bảo luôn tồn tại cách cắt thỏa mãn.

Example

Input:

5 5 3 2

Output:

2

Input:

7 5 5 2

Output:

2

```
1. class CF {
2.     public static void main(String[] args) {
3.         Scanner sc = new Scanner(System.in);
4.         int n = sc.nextInt(), a[] = new int[3], max = 0;
5.         for (int i = 0; i < 3; i++) a[i] = sc.nextInt();
6.         Arrays.sort(a);
7.         for (int i = 0; i <= n / a[2]; i++)
8.             for (int j = 0; j <= (n - a[2] * i) / a[1]; j++) {
9.                 int k = (n - a[2] * i - a[1] * j) / a[0];
10.                  if (a[0] * k + a[1] * j + a[2] * i == n && i +
   j + k > max) max = i + j + k;
11.             }
12.         System.out.println(max);
13.     }
14. }
```

5. Một cái túi

Cho trước một tập gồm n đồ vật, mỗi đồ vật có một chi phí Vi và một giá trị i,xác định xem cần chọn những đồ vật nào

sao cho tổng chi phí nhỏ hơn một ngưỡng cho trước (giới hạn của balô) và tổng giá trị cao nhất có thể được.

Input

Dòng 1: n, S (với S là giới hạn của balô)

n dòng tiếp theo: Mỗi dòng gồm 2 số Vi, Wi là chi phí và giá trị đồ vật thứ i.

Output

Dòng 1: Ghi 2 số nguyên $, k là tổng giá trị lớn nhất tìm được và số đồ vật được chọn

k dòng tiếp theo ghi chỉ số các đồ vật được chọn

```
public class BaiToanMotCaiTui {


  static int KQ[];
  static int vt[];


  static void SX(int w[], int v[], int n) {
     for (int i = 0; i < n; i++) {
        vt[i] = i + 1;
     }
     int i, j, temp;
     for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - 1 - i; j++) {
           if (w[j] < w[j + 1]) {
              temp = vt[j];
              vt[j] = vt[j + 1];
              vt[j + 1] = temp;
```

```
                temp = w[j];
                w[j] = w[j + 1];
                w[j + 1] = temp;
                temp = v[j];
                v[j] = v[j + 1];
                v[j + 1] = temp;
            }
        }
    }
}

static void Push(int w[], int v[], int n, int s) {
    SX(w, v, n);
    int i = 0, count = 0, giatri = 0, chiphi = 0;
    while (chiphi < s && i < n) {
        if (chiphi + v[i] <= s) {
            KQ[count++] = vt[i];
            chiphi += v[i];
            giatri += w[i];
        }
        i++;
    }
    System.out.println(giatri + " " + count);
    for (i = 0; i < count; i++) {
        System.out.println(KQ[i]);
```

```
    }
  }


  public static void main(String[] args) {
    int n, s;
    int w[], v[];
    Scanner sc = new Scanner(System.in);
    n = sc.nextInt();
    s = sc.nextInt();
    v = new int[n];
    w = new int[n];
    for (int i = 0; i < n; i++) {
      v[i] = sc.nextInt();
      w[i] = sc.nextInt();
    }
    KQ = new int[n];
    vt = new int[n];
    Push(w, v, n, s);
  }
```

6. Đặt lại xâu kí tự

Cho xâu ký tự s[] độ dài n và số tự nhiên d. Hãy sắp đặt lại các ký tự

trong xâu s[] sao cho hai ký tự giống nhau đều cách nhau một khoảng là d.

Nếu bài toán có nhiều nghiệm, hãy đưa ra một cách sắp đặt đầu tiên tìm được.

Nếu bài toán không có lời giải hãy đưa ra thông báo “Vô nghiệm”.

```
public class DatLaiXauKyTu {

   static String s;

   static String st;

   static int d,freq[];

   static void Get_Freq() {

      st = "";

      for (int i = 0; i < s.length(); i++) {

         String t = s.substring(i, i + 1);

         if (!st.contains(t)) {

            st += t;

         } else {

            freq[st.indexOf(t)]++;

         }

      }

   }


   static void bubble_sort() {

      int i, j, temp;

      char sh[] = st.toCharArray();

      int n=st.length();

      for (i = 0; i < n - 1; i++) {

         for (j = 0; j < n - 1 - i; j++) {

            if (freq[j] < freq[j + 1]) {

               temp = freq[j];
```

```
                freq[j] = freq[j + 1];
                freq[j + 1] = temp;
                char tmp = sh[j];
                sh[j] = sh[j + 1];
                sh[j + 1] = tmp;
            }
        }
    }
    st="";
    for(i=0;i<n;i++)st+=sh[i];

}

static void Greedy_Arrang_String() {
    Get_Freq();
    bubble_sort();
    int i=0;
    int k=st.length();
    int n=s.length();
    char sh[]=new char[n];
    while(i<k){
        int p=freq[0];
        for(int t=0;t<p;t++){
            if(i+(t*d)<n){
                sh[i+(t*d)]=st.charAt(i);
```

```
            }else{
                System.out.println("Vo Nghiem");
            }
        }
        i++;
    }
    System.out.print("KQ: ");
    for(i=0;i<n;i++)System.out.print(sh[i]);

}

public static void main(String[] args) {
    s = "ABB";
    d=2;
    st = "";
    freq = new int[s.length()];
    for (int i = 0; i < s.length(); i++) {
        freq[i] = 1;
    }
    Greedy_Arrang_String();
}
}
```

7. Lựa chọn hành động

Cho tập gồm n hành động, mỗi hành động được biểu diễn như bộ đôi thời gian bắt đầu si và thời gian kết thúc fi (i=1, 2, .., n). Bài toán đặt ra là hãy

lựa chọn nhiều nhất các hành động có thể thực hiện bởi một máy hoặc một cá nhân mà không xảy ra tranh chấp. Giả sử mỗi hành động chỉ thực hiện đơn lẻ tại một thời điểm.

Input:

- Số lượng hành động: 6
- Thời gian bắt đầu Start []
- Thời gian kết thúc Finish[]

Output: Số lượng lớn nhất các hành động có thể thực hiện bởi một người.

OPT[]

```
public class LuaChonHanhDong {


  //Thuật toán Greedy-ActivitiesN-Selection
  static void GAS(int s[], int f[], int n) {
     int i, j,count;
     int opt[]=new int[n];
     int N[] = new int[n];
     for (i = 0; i < n; i++) {
        N[i] = 1;
     }
     i=0;
     count=0;
     N[0] = 0;
     opt[0]=0;
     for (j = 0; j < n; j++) {
```

```
        if (N[j] == 1 && s[j] >= f[i]) {
            opt[++count]=j;
            i = j;
            N[i] = 0;
          }
      }
      for(i=0;i<=count;i++)System.out.print((opt[i]+1)+" ");
   }

   public static void main(String[] args) {
      int n = 8;
      int s[] = {1, 3, 0, 5, 8, 5, 9, 14};
      int f[] = {2, 4, 6, 7, 9, 9, 12, 18};
      GAS(s, f, n);
   }
}
```

8. N_Ropes

Cho n dây với chiều dài khác nhau. Ta cần phải

nối các dây lại với nhau thành một dây. Chi phí nối hai dây lại với nhau

được tính bằng tổng độ dài hai dây. Nhiệm vụ của bài toán là tìm cách

nối các dây lại với nhau thành một dây sao cho chi phí nối các dây lại với

nhau là ít nhất.

Input:

- Số lượng dây:

- Độ dài dây

Output: Chi phí nối dây nhỏ nhất.

OPT

```
public class N_Ropes {


   static class PriQueue {


      private int size;

      private int a[] = new int[1000];


      public int Size() {

         return size;

      }


      public boolean isEmpty() {

         return size == 0;

      }


      public void push(int x) {

         int i = size-1;

         size++;

         while (i>=0 && a[i]>x) {

            a[i+1]=a[i];

            i--;

         }

         a[i+1] = x;
```

```
    }

    public int pop() {
        if (!this.isEmpty()) {
            int item = a[0];
            for (int i = 0; i < size - 1; i++) {
                a[i] = a[i + 1];
            }
            size--;
            return item;
        }
        return 0;
    }

    public int top() {
        if (!this.isEmpty()) {
            return a[0];
        }
        return -1;
    }
}

static void Greedy_N_Ropes(int L[], int n) {
    PriQueue qu = new PriQueue();
    for (int i = 0; i < n; i++) {
```

```
        qu.push(L[i]);
      }
      int opt = 0;
      while (qu.size > 1) {
        int chiphi = qu.pop()+qu.pop();
        opt += (chiphi);
        qu.push(chiphi);
      }
   }

   public static void main(String[] args) {
      int n = 4;
      int L[] = {4, 3, 2, 6};
      Greedy_N_Ropes(L, n);
   }
}
```

9. Người du lịch

Đầu vào: số thành phố n, chi phí tử thành phố i đến thành phố j (C[i][j])

Đầu ra: Hành trình tối ưu và chi phi tương ứng.

```
public class NguoiDuLich {


   static void ThamLam(int c[][], int n) {
      int kq[] = new int[n + 1];
      int kt[] = new int[n + 1];
```

```
int i, j, chiphi, vitri;
for (i = 1; i < n; i++) {
   kt[i] = 0;
}
chiphi = 0;
vitri = 0;
kt[0] = 1;
kq[0] = 0;
for (i = 1; i < n; i++) {
   int min = Integer.MAX_VALUE;
   int vt=0;
   for (j = 1; j < n; j++) {
      if (kt[j] == 0 && c[vitri][j] < min) {
         min = c[vitri][j];
         vt=j;
      }
   }
   kq[i] = vt;
   kt[vt] = 1;
   chiphi += min;
   vitri = vt;
}
kq[n] = 0;
chiphi += c[vitri][0];
System.out.println("chu trinh:");
```

```
        for(i=0;i<=n;i++)System.out.print(kq[i]+" ");
        System.out.println("\n Chi phi:"+chiphi);
    }

    public static void main(String[] args) {
        int n;
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        int c[][] = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                c[i][j] = sc.nextInt();
            }
        }
        ThamLam(c, n);
    }
}
```