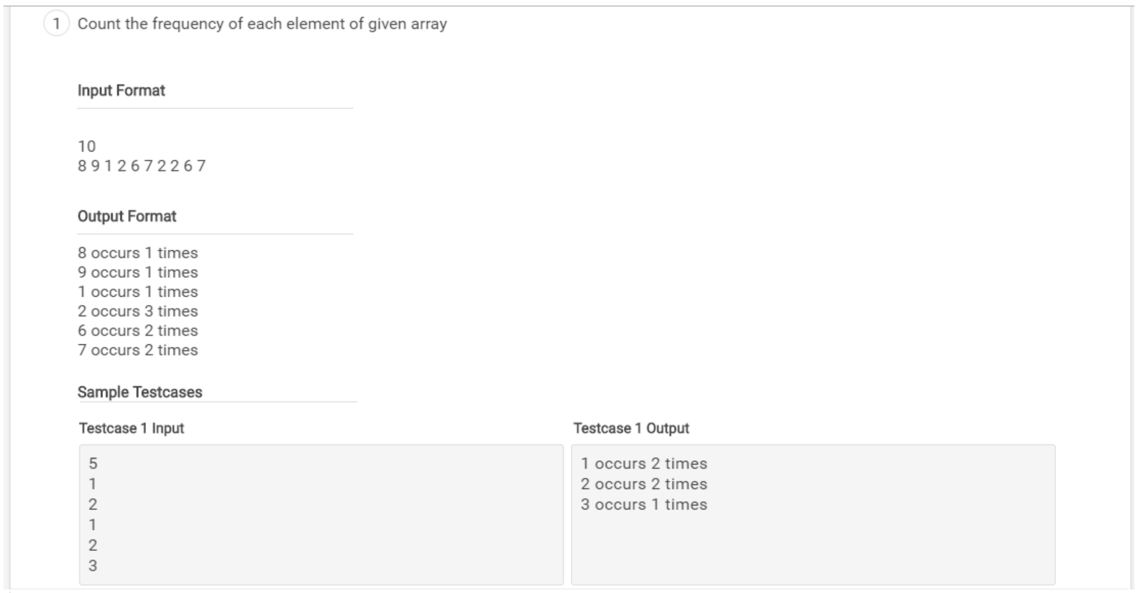


1 Count the frequency of each element of given array

Input Format

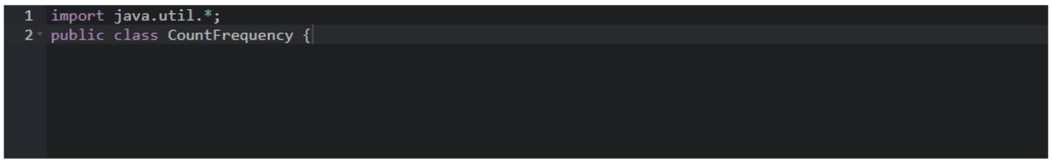
```
10
8 9 1 2 6 7 2 2 6 7
```

Output Format

```
8 occurs 1 times
9 occurs 1 times
1 occurs 1 times
2 occurs 3 times
6 occurs 2 times
7 occurs 2 times
```

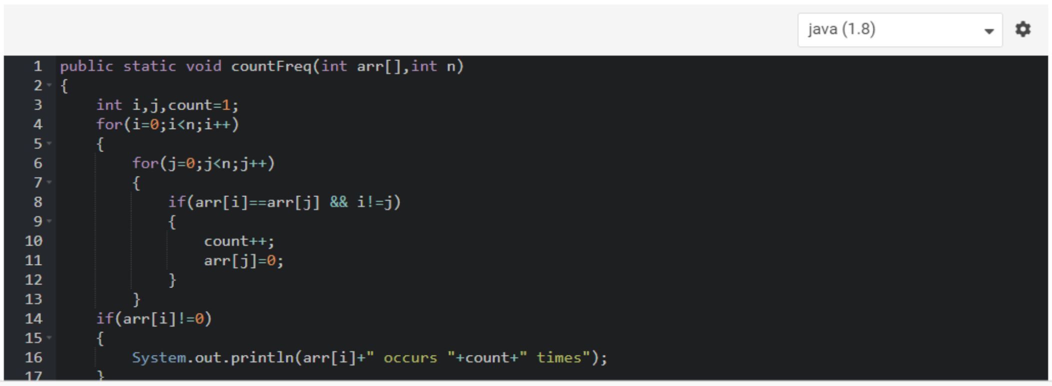
Sample Testcases

Testcase 1 Input	Testcase 1 Output
5 1 2 1 2 3	1 occurs 2 times 2 occurs 2 times 3 occurs 1 times



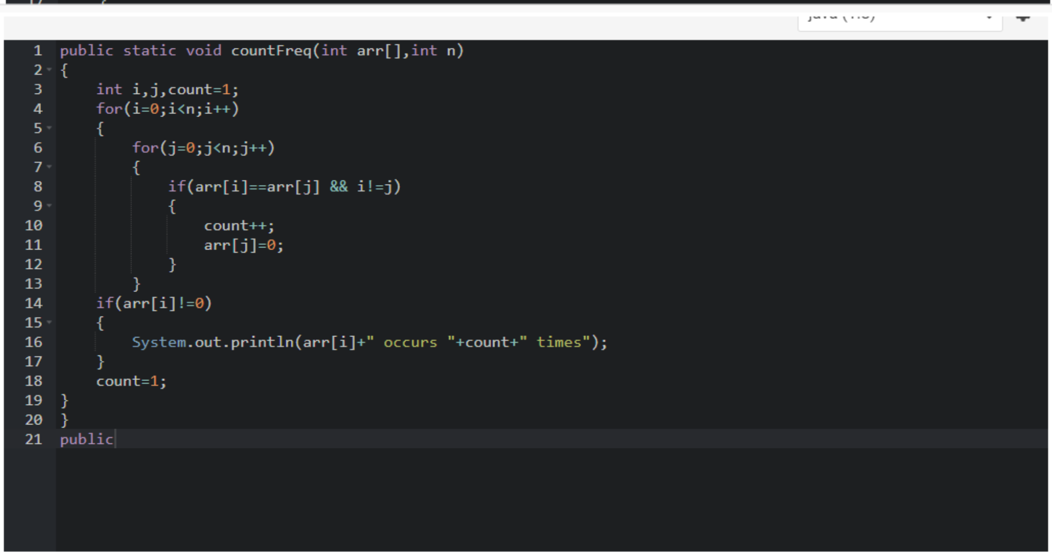
```java
import java.util.*;
public class CountFrequency {
```

Fill the remaining code here



java (1.8)

```java
public static void countFreq(int arr[],int n)
{
    int i,j,count=1;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(arr[i]==arr[j] && i!=j)
            {
                count++;
                arr[j]=0;
            }
        }
    if(arr[i]!=0)
    {
        System.out.println(arr[i]+" occurs "+count+" times");
    }
```



```java
public static void countFreq(int arr[],int n)
{
    int i,j,count=1;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(arr[i]==arr[j] && i!=j)
            {
                count++;
                arr[j]=0;
            }
        }
    if(arr[i]!=0)
    {
        System.out.println(arr[i]+" occurs "+count+" times");
    }
    count=1;
}
}
public
```

```java
    if(arr[i]!=0)
    {
        System.out.println(arr[i]+" occurs "+count+" times");
    }
    count=1;
}
}
public
```

Footer Snippet

```java
    static void main(java.lang.String a[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int arr[]=new int[n];
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }
        countFreq(arr,n);
    }
}
```

2 Given two strings S and A. Print "1" if both strings are anagrams otherwise print "0".

Input Format

First line of input contains a single integer T which denotes the number of test cases. T test cases follows, first line of each test case contains a string S. Second and last line of each test case consists of string A.

```
2
cdbkdub
dsbkcsdn
hello
elhlo
```

Output Format

Corresponding to each test case, print the required output.

```
0
1
```

Constraints

1<=T<=100
1<= length(S and A) <=1000

Output Format

Corresponding to each test case, print the required output.

```
0
1
```

Constraints

1<=T<=100
1<= length(S and A) <=1000

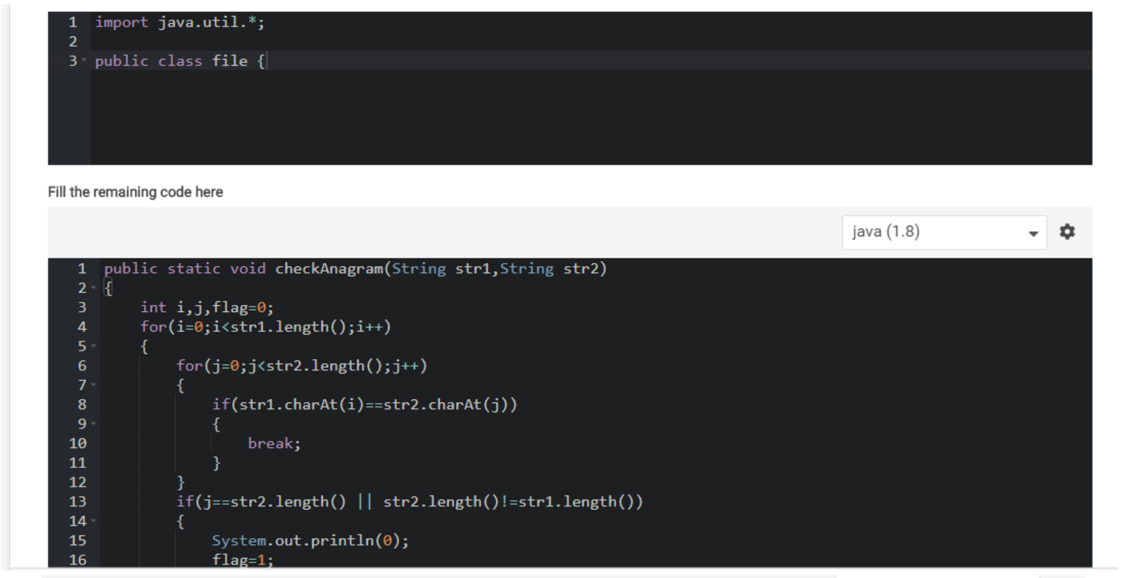
Sample Testcases

Testcase 1 Input	Testcase 1 Output
2 cdbkdub dsbkcsdn hello elhlo	0 1

Time Limit: 0 ms **Memory Limit:** 0 kb **Code Size:** 0 kb

✓ Marks: 5 ✗ Negative Marks: 0

Mark as error

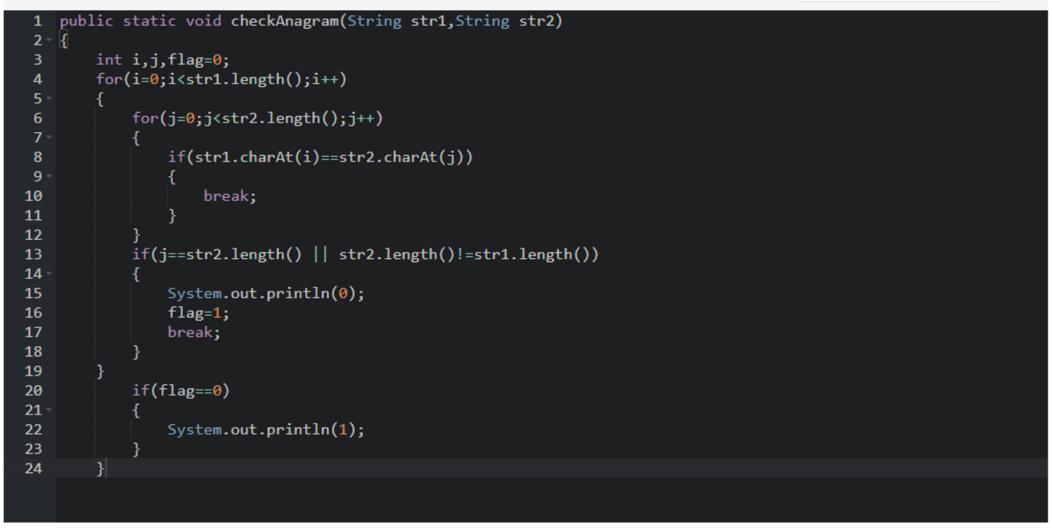


```java
import java.util.*;

public class file {
```

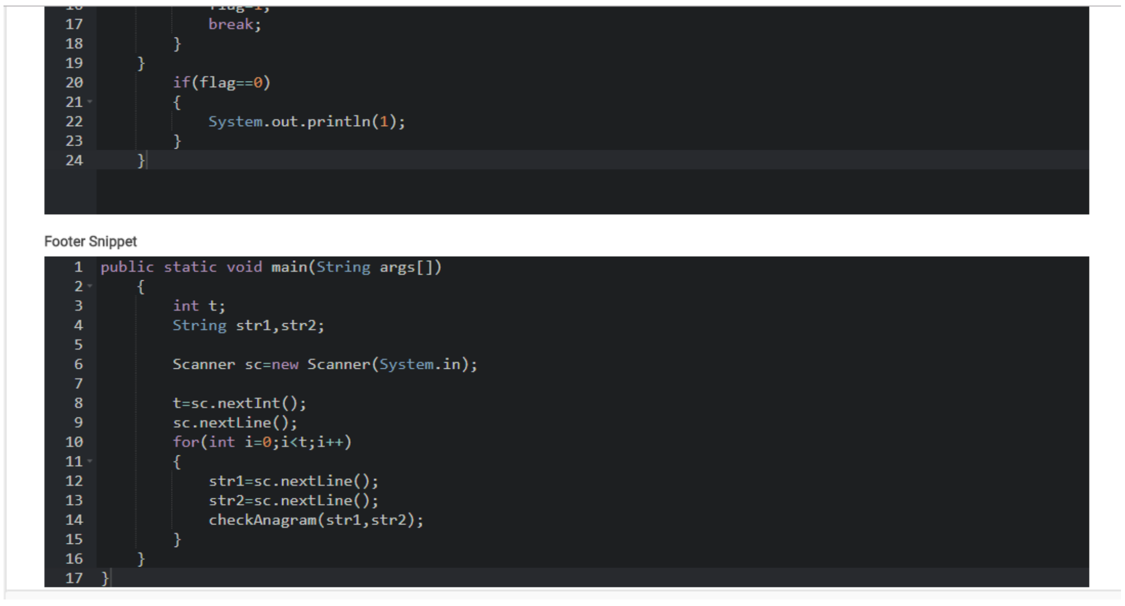
Fill the remaining code here

java (1.8)

```java
public static void checkAnagram(String str1,String str2)
{
    int i,j,flag=0;
    for(i=0;i<str1.length();i++)
    {
        for(j=0;j<str2.length();j++)
        {
            if(str1.charAt(i)==str2.charAt(j))
            {
                break;
            }
        }
        if(j==str2.length() || str2.length()!=str1.length())
        {
            System.out.println(0);
            flag=1;
```



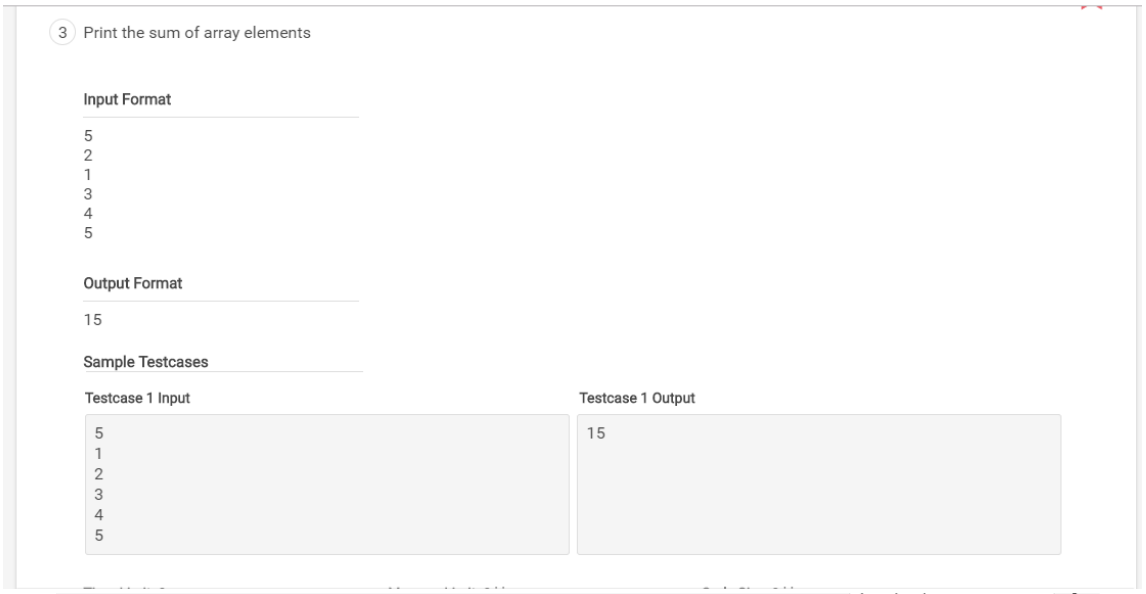
```java
public static void checkAnagram(String str1,String str2)
{
    int i,j,flag=0;
    for(i=0;i<str1.length();i++)
    {
        for(j=0;j<str2.length();j++)
        {
            if(str1.charAt(i)==str2.charAt(j))
            {
                break;
            }
        }
        if(j==str2.length() || str2.length()!=str1.length())
        {
            System.out.println(0);
            flag=1;
            break;
        }
    }
        if(flag==0)
        {
            System.out.println(1);
        }
    }
```



```java
            break;
        }
    }
        if(flag==0)
        {
            System.out.println(1);
        }
    }
```

Footer Snippet

```java
public static void main(String args[])
    {
        int t;
        String str1,str2;

        Scanner sc=new Scanner(System.in);

        t=sc.nextInt();
        sc.nextLine();
        for(int i=0;i<t;i++)
        {
            str1=sc.nextLine();
            str2=sc.nextLine();
            checkAnagram(str1,str2);
        }
    }
}
```



3 Print the sum of array elements

Input Format

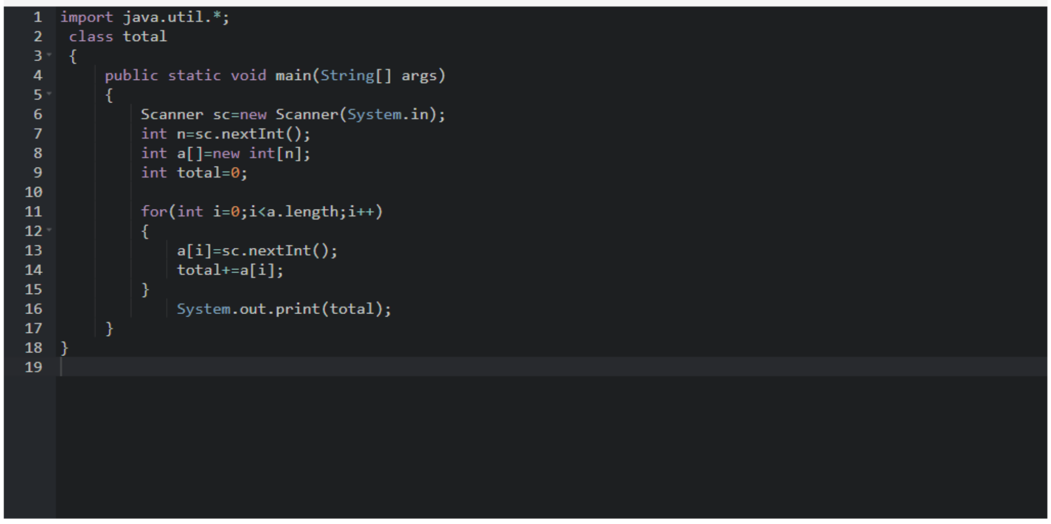
5
2
1
3
4
5

Output Format

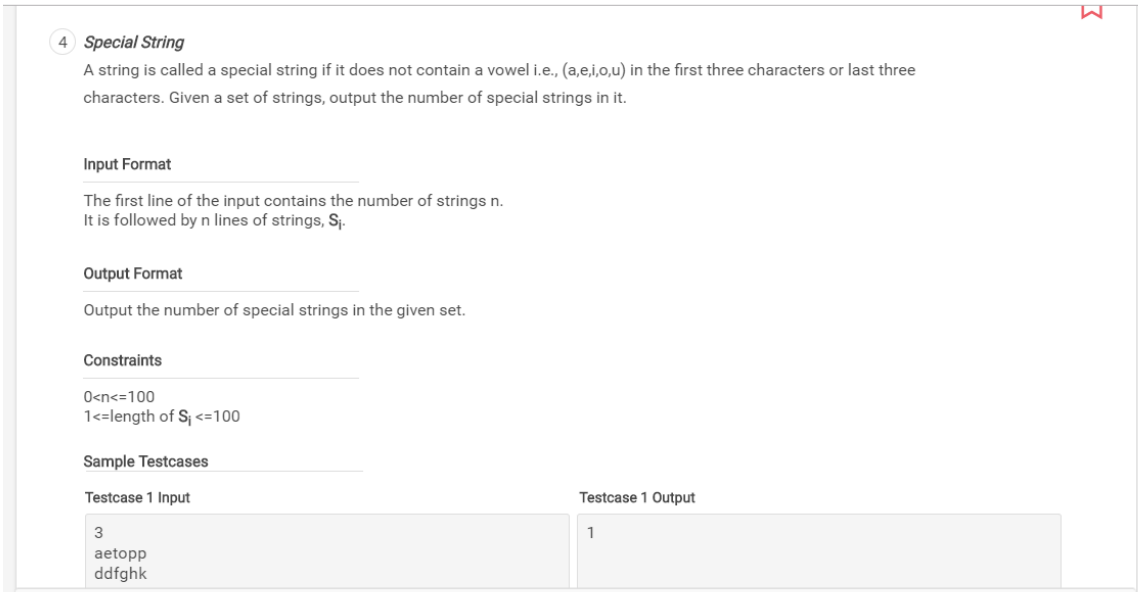
15

Sample Testcases

Testcase 1 Input	Testcase 1 Output
5 1 2 3 4 5	15



```java
import java.util.*;
 class total
 {
     public static void main(String[] args)
     {
         Scanner sc=new Scanner(System.in);
         int n=sc.nextInt();
         int a[]=new int[n];
         int total=0;

         for(int i=0;i<a.length;i++)
         {
             a[i]=sc.nextInt();
             total+=a[i];
         }
             System.out.print(total);
     }
}

```



4 *Special String*

A string is called a special string if it does not contain a vowel i.e., (a,e,i,o,u) in the first three characters or last three characters. Given a set of strings, output the number of special strings in it.

Input Format

The first line of the input contains the number of strings n.
It is followed by n lines of strings, S_i.

Output Format

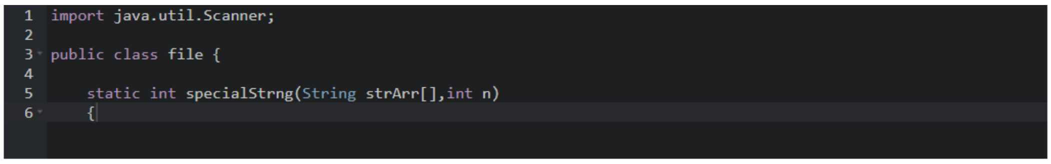
Output the number of special strings in the given set.

Constraints

0<n<=100
1<=length of S_i <=100

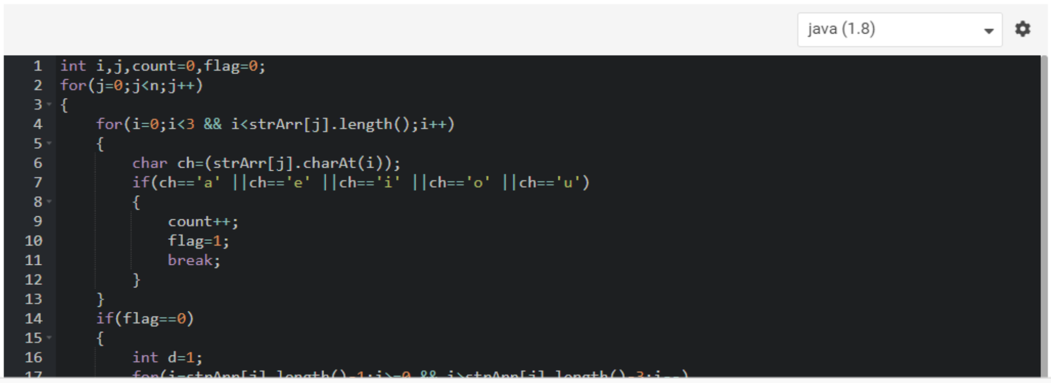
Sample Testcases

Testcase 1 Input	Testcase 1 Output
3 aetopp ddfghk	1



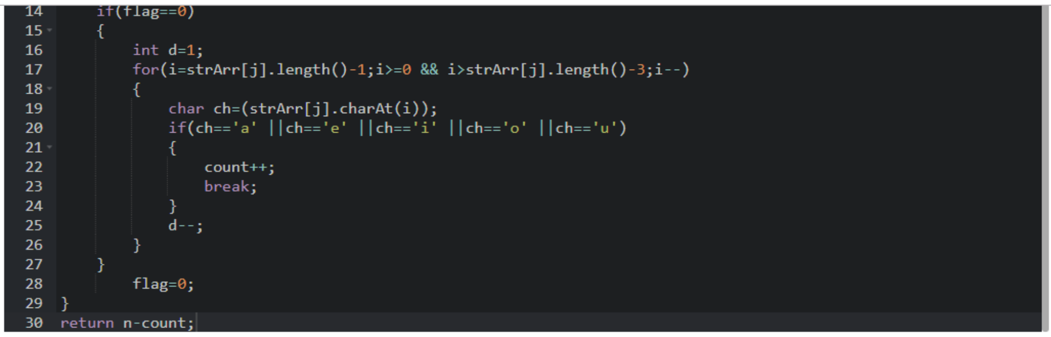
```java
import java.util.Scanner;

public class file {

    static int specialStrng(String strArr[],int n)
    {
```

Fill the remaining code here



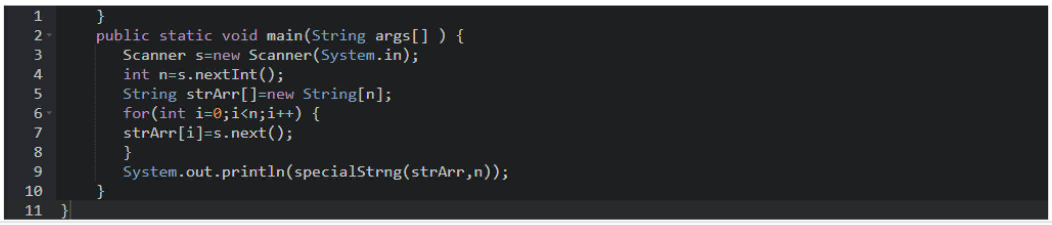
java (1.8)

```java
int i,j,count=0,flag=0;
for(j=0;j<n;j++)
{
    for(i=0;i<3 && i<strArr[j].length();i++)
    {
        char ch=(strArr[j].charAt(i));
        if(ch=='a' ||ch=='e' ||ch=='i' ||ch=='o' ||ch=='u')
        {
            count++;
            flag=1;
            break;
        }
    }
    if(flag==0)
    {
        int d=1;
```



```java
    if(flag==0)
    {
        int d=1;
        for(i=strArr[j].length()-1;i>=0 && i>strArr[j].length()-3;i--)
        {
            char ch=(strArr[j].charAt(i));
            if(ch=='a' ||ch=='e' ||ch=='i' ||ch=='o' ||ch=='u')
            {
                count++;
                break;
            }
            d--;
        }
    }
        flag=0;
}
return n-count;
```

Footer Snippet



```java
    }
    public static void main(String args[] ) {
       Scanner s=new Scanner(System.in);
       int n=s.nextInt();
       String strArr[]=new String[n];
       for(int i=0;i<n;i++) {
       strArr[i]=s.next();
       }
       System.out.println(specialStrng(strArr,n));
    }
}
```

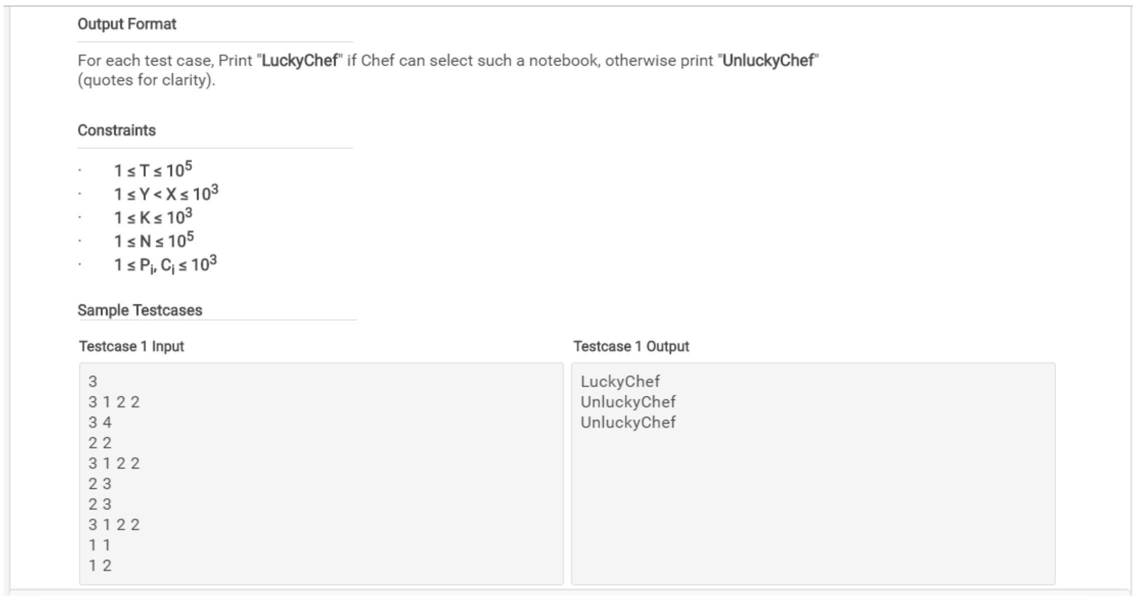
6 Chef likes to write poetry. Today, he has decided to write a **X** pages long poetry, but unfortunately his notebook has only **Y** pages left in it. Thus he decided to buy a new CHEFMATE notebook and went to the stationary shop. Shopkeeper showed him some **N** notebooks, where the number of pages and price of the ith one are P_i pages and C_i rubles respectively. Chef has spent some money preparing for Ksen's birthday, and then he has only **K** rubles left for now.
Chef wants to buy a single notebook such that the price of the notebook should not exceed his budget and he is able to complete his poetry.
Help Chef accomplishing this task. You just need to tell him whether he can buy such a notebook or not. Note that Chef can use all of the **Y** pages in the current notebook, and Chef can buy only one notebook because Chef doesn't want to use many notebooks.

Input Format

The first line of input contains an integer **T**, denoting the number of test cases. Then **T** test cases are follow.
The first line of each test case contains four space-separated integers **X**, **Y**, **K** and **N**, described in the statement. The ith line of the next **N** lines contains two space-separated integers P_i and C_i, denoting the number of pages and price of the ithnotebook respectively.

Output Format

For each test case, Print "**LuckyChef**" if Chef can select such a notebook, otherwise print "**UnluckyChef**" (quotes for clarity).



Output Format

For each test case, Print "**LuckyChef**" if Chef can select such a notebook, otherwise print "**UnluckyChef**" (quotes for clarity).

Constraints

- $1 \le T \le 10^5$
- $1 \le Y < X \le 10^3$
- $1 \le K \le 10^3$
- $1 \le N \le 10^5$
- $1 \le P_i, C_i \le 10^3$

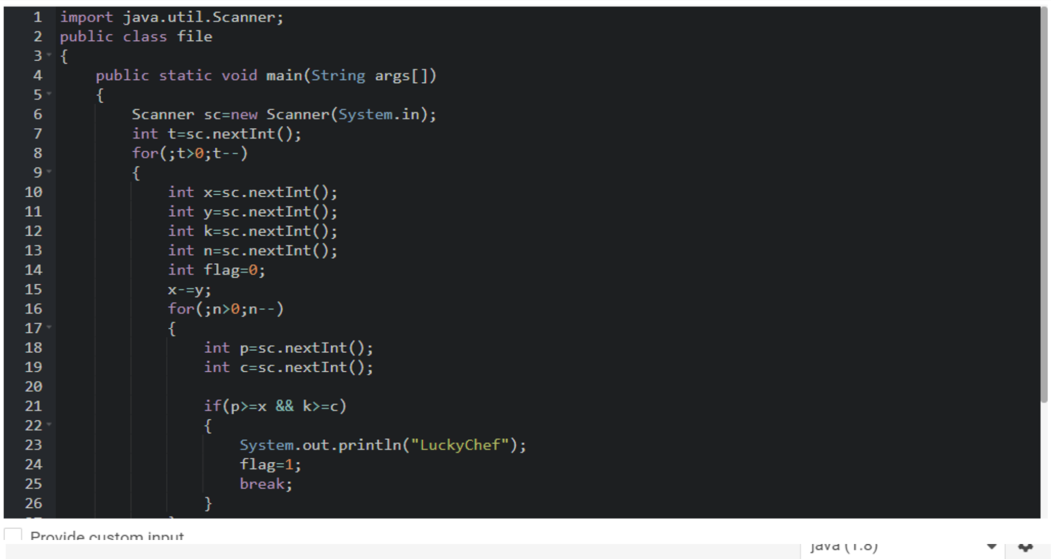
Sample Testcases

Testcase 1 Input

```
3
3 1 2 2
3 4
2 2
3 1 2 2
2 3
2 3
3 1 2 2
1 1
1 2
```

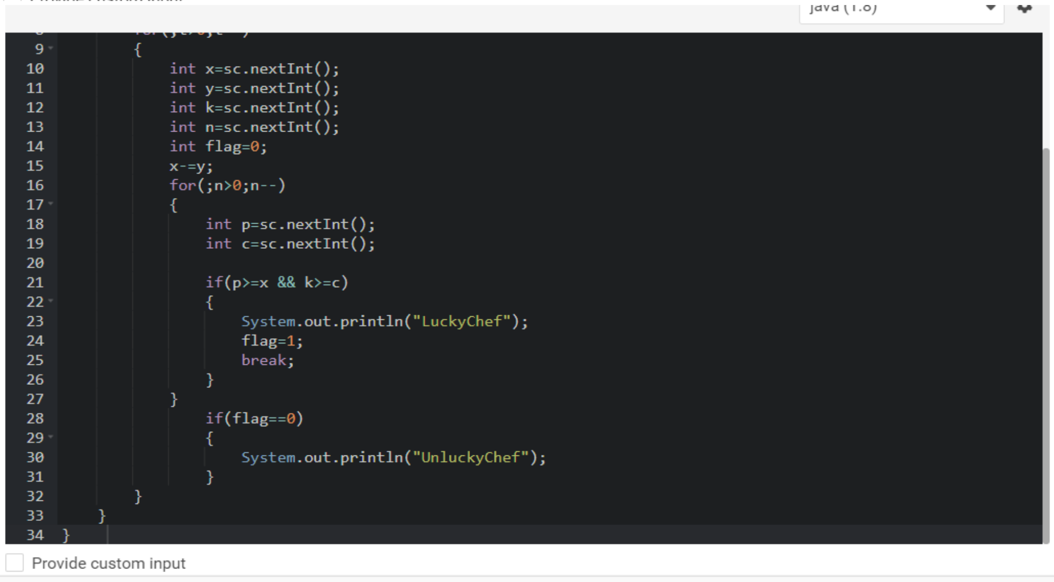
Testcase 1 Output

```
LuckyChef
UnluckyChef
UnluckyChef
```



```java
import java.util.Scanner;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int t=sc.nextInt();
        for(;t>0;t--)
        {
            int x=sc.nextInt();
            int y=sc.nextInt();
            int k=sc.nextInt();
            int n=sc.nextInt();
            int flag=0;
            x-=y;
            for(;n>0;n--)
            {
                int p=sc.nextInt();
                int c=sc.nextInt();

                if(p>=x && k>=c)
                {
                    System.out.println("LuckyChef");
                    flag=1;
                    break;
                }
```

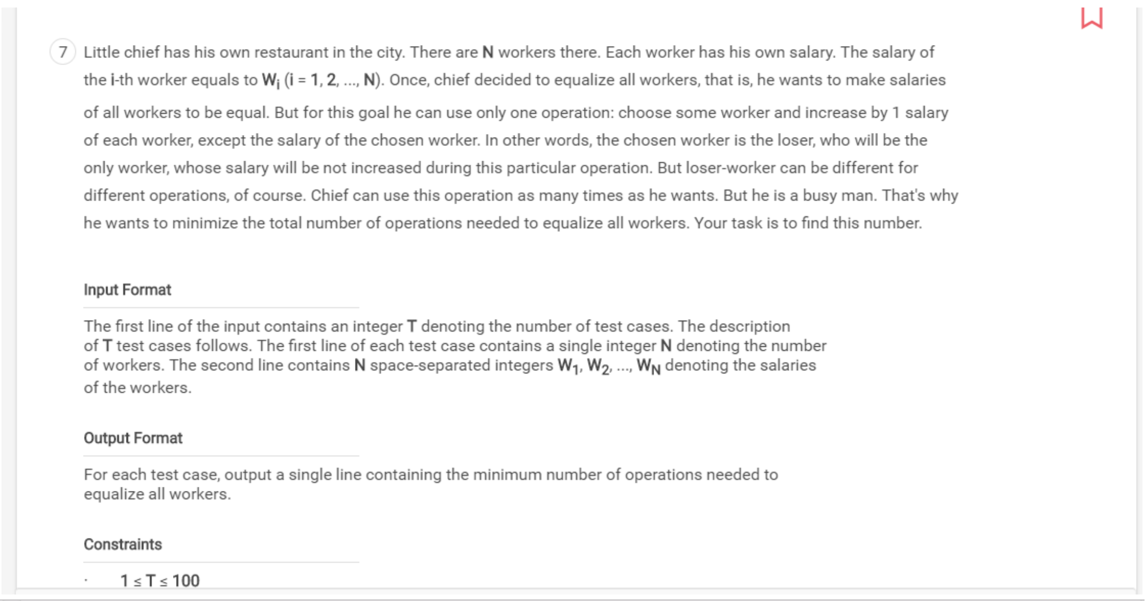
Provide custom input



java (1.8)

```java
        {
            int x=sc.nextInt();
            int y=sc.nextInt();
            int k=sc.nextInt();
            int n=sc.nextInt();
            int flag=0;
            x-=y;
            for(;n>0;n--)
            {
                int p=sc.nextInt();
                int c=sc.nextInt();

                if(p>=x && k>=c)
                {
                    System.out.println("LuckyChef");
                    flag=1;
                    break;
                }
            }
                if(flag==0)
                {
                    System.out.println("UnluckyChef");
                }
        }
    }
}
```

Provide custom input



7 Little chief has his own restaurant in the city. There are **N** workers there. Each worker has his own salary. The salary of the **i**-th worker equals to W_i (**i** = 1, 2, ..., **N**). Once, chief decided to equalize all workers, that is, he wants to make salaries of all workers to be equal. But for this goal he can use only one operation: choose some worker and increase by 1 salary of each worker, except the salary of the chosen worker. In other words, the chosen worker is the loser, who will be the only worker, whose salary will be not increased during this particular operation. But loser-worker can be different for different operations, of course. Chief can use this operation as many times as he wants. But he is a busy man. That's why he wants to minimize the total number of operations needed to equalize all workers. Your task is to find this number.

Input Format

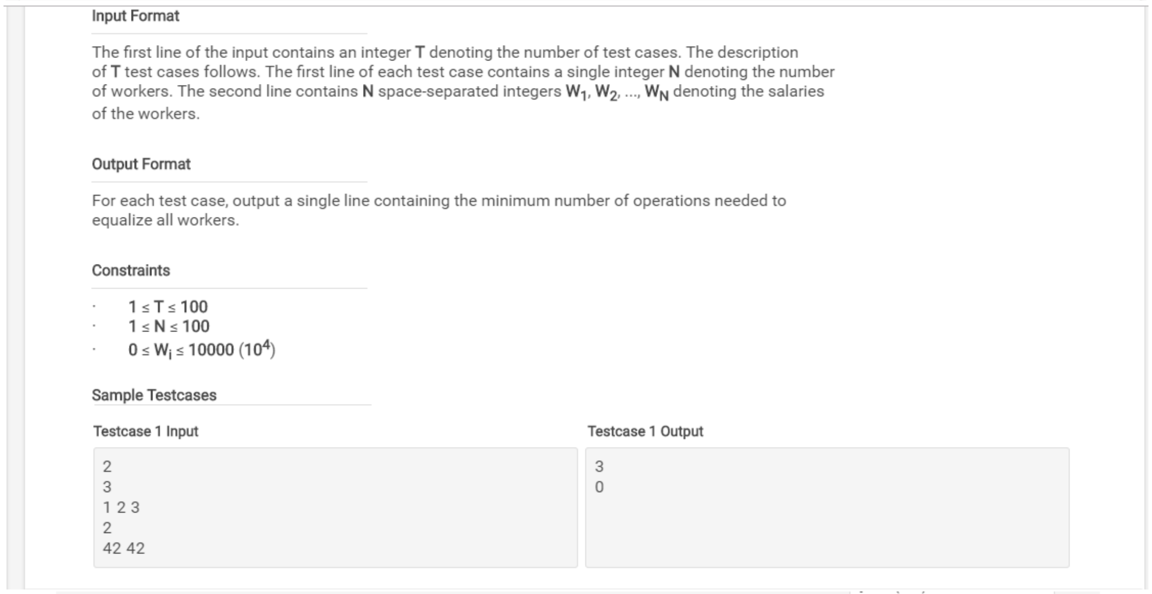
The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows. The first line of each test case contains a single integer **N** denoting the number of workers. The second line contains **N** space-separated integers W_1, W_2, ..., W_N denoting the salaries of the workers.

Output Format

For each test case, output a single line containing the minimum number of operations needed to equalize all workers.

Constraints

- $1 \le T \le 100$



Input Format

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows. The first line of each test case contains a single integer **N** denoting the number of workers. The second line contains **N** space-separated integers W_1, W_2, ..., W_N denoting the salaries of the workers.

Output Format

For each test case, output a single line containing the minimum number of operations needed to equalize all workers.

Constraints

- $1 \le T \le 100$
- $1 \le N \le 100$
- $0 \le W_i \le 10000$ (10^4)

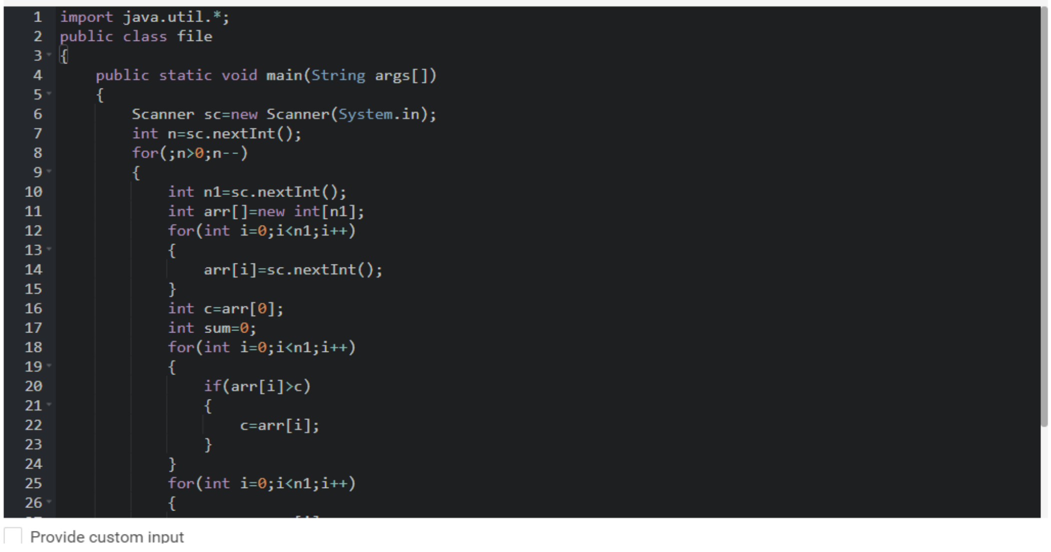
Sample Testcases

Testcase 1 Input

```
2
3
1 2 3
2
42 42
```

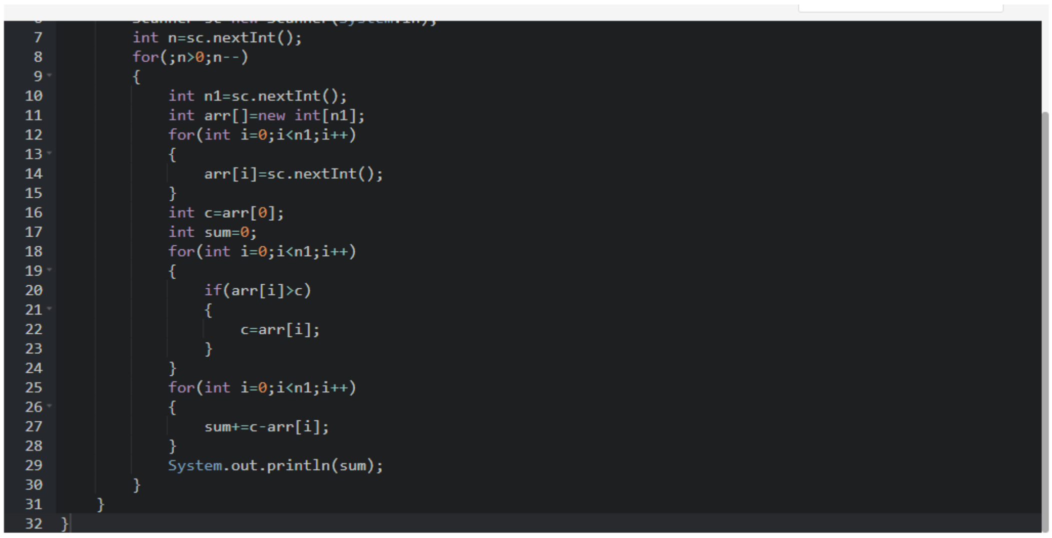
Testcase 1 Output

```
3
0
```



```
import java.util.*;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        for(;n>0;n--)
        {
            int n1=sc.nextInt();
            int arr[]=new int[n1];
            for(int i=0;i<n1;i++)
            {
                arr[i]=sc.nextInt();
            }
            int c=arr[0];
            int sum=0;
            for(int i=0;i<n1;i++)
            {
                if(arr[i]>c)
                {
                    c=arr[i];
                }
            }
            for(int i=0;i<n1;i++)
            {
```

Provide custom input


```
        int n=sc.nextInt();
        for(;n>0;n--)
        {
            int n1=sc.nextInt();
            int arr[]=new int[n1];
            for(int i=0;i<n1;i++)
            {
                arr[i]=sc.nextInt();
            }
            int c=arr[0];
            int sum=0;
            for(int i=0;i<n1;i++)
            {
                if(arr[i]>c)
                {
                    c=arr[i];
                }
            }
            for(int i=0;i<n1;i++)
            {
                sum+=c-arr[i];
            }
            System.out.println(sum);
        }
    }
}
```

Provide custom input

8 Forgotten languages (also known as extinct languages) are languages that are no longer in use. Such languages were, probably, widely used before and no one could have ever imagined that they will become extinct at some point. Unfortunately, that is what happened to them. On the happy side of things, a language may be dead, but some of its words may continue to be used in other languages.

Using something called as *the Internet*, you have acquired a dictionary of **N** words of a forgotten language. Meanwhile, you also know **K** phrases used in modern languages. For each of the words of the forgotten language, your task is to determine whether the word is still in use in any of these **K** modern phrases or not.

Input Format

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.
The first line of a test case description contains two space separated positive integers **N** and **K**.
The second line of the description contains **N** strings denoting a dictionary of the forgotten language.
Each of the next **K** lines of the description starts with one positive integer **L** denoting the number of words in the corresponding phrase in modern languages. The integer is followed by **L** strings (not necessarily distinct) denoting the phrase.

Output Format

For each test case, output a single line containing **N** tokens (space-separated): if the ith word of the dictionary exists in at least one phrase in modern languages, then you should output **YES** as the ith token, otherwise **NO**

necessarily distinct) denoting the phrase.

Output Format

For each test case, output a single line containing **N** tokens (space-separated): if the ith word of the dictionary exists in at least one phrase in modern languages, then you should output **YES** as the ith token, otherwise **NO**

Constraints

- 1 ≤ **T** ≤ 20
- 1 ≤ **N** ≤ 100
- 1 ≤ **K, L** ≤ 50
- 1 ≤ length of any string in the input ≤ 5

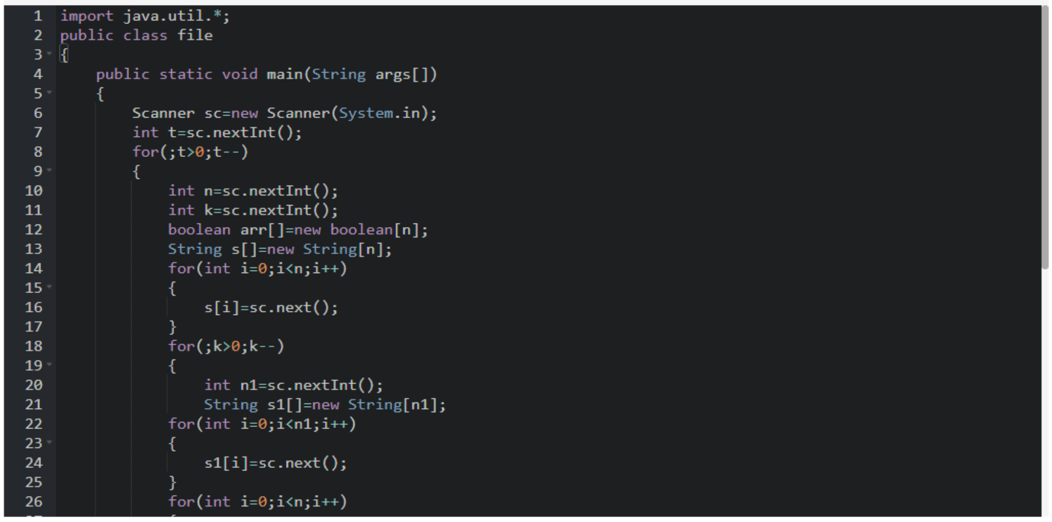
Sample Testcases

Testcase 1 Input

```
2
3 2
piygu ezyfo rzotm
1 piygu
6 tefwz tefwz piygu ezyfo tefwz piygu
4 1
kssdy tjzhy ljzym kegqz
4 kegqz kegqz kegqz vxvyj
```

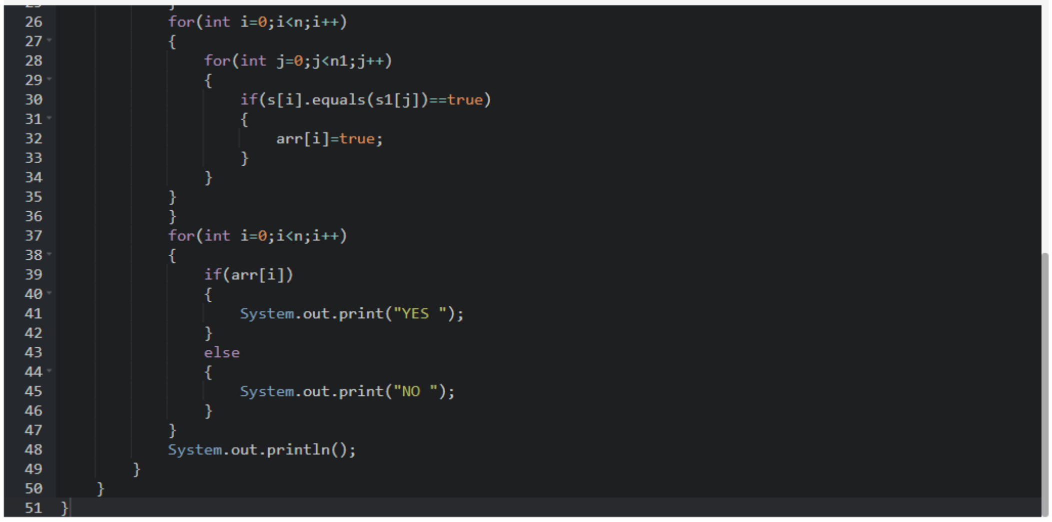
Testcase 1 Output

```
YES YES NO
NO NO NO YES
```



```java
import java.util.*;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int t=sc.nextInt();
        for(;t>0;t--)
        {
            int n=sc.nextInt();
            int k=sc.nextInt();
            boolean arr[]=new boolean[n];
            String s[]=new String[n];
            for(int i=0;i<n;i++)
            {
                s[i]=sc.next();
            }
            for(;k>0;k--)
            {
                int n1=sc.nextInt();
                String s1[]=new String[n1];
            for(int i=0;i<n1;i++)
            {
                s1[i]=sc.next();
            }
            for(int i=0;i<n;i++)
```

Provide custom input



```java
            for(int i=0;i<n;i++)
            {
                for(int j=0;j<n1;j++)
                {
                    if(s[i].equals(s1[j])==true)
                    {
                        arr[i]=true;
                    }
                }
            }
            }
            for(int i=0;i<n;i++)
            {
                if(arr[i])
                {
                    System.out.print("YES ");
                }
                else
                {
                    System.out.print("NO ");
                }
            }
            System.out.println();
        }
    }
}
```

Provide custom input

9 John Watson knows of an operation called a *right circular rotation* on an array of integers. One rotation operation moves the last array element to the first position and shifts all remaining elements right one. To test Sherlock's abilities, Watson provides Sherlock with an array of integers. Sherlock is to perform the rotation operation a number of times then determine the value of the element at a given position.

For each array, perform a number of right circular rotations and return the value of the element at a given index.

For example, array , a=[3,4,5] number of rotations, k=2 and indices to check, m=[1,2] .

First we perform the two rotations:

[3,4,5] -> [5,3,4] ->[4,5,3]

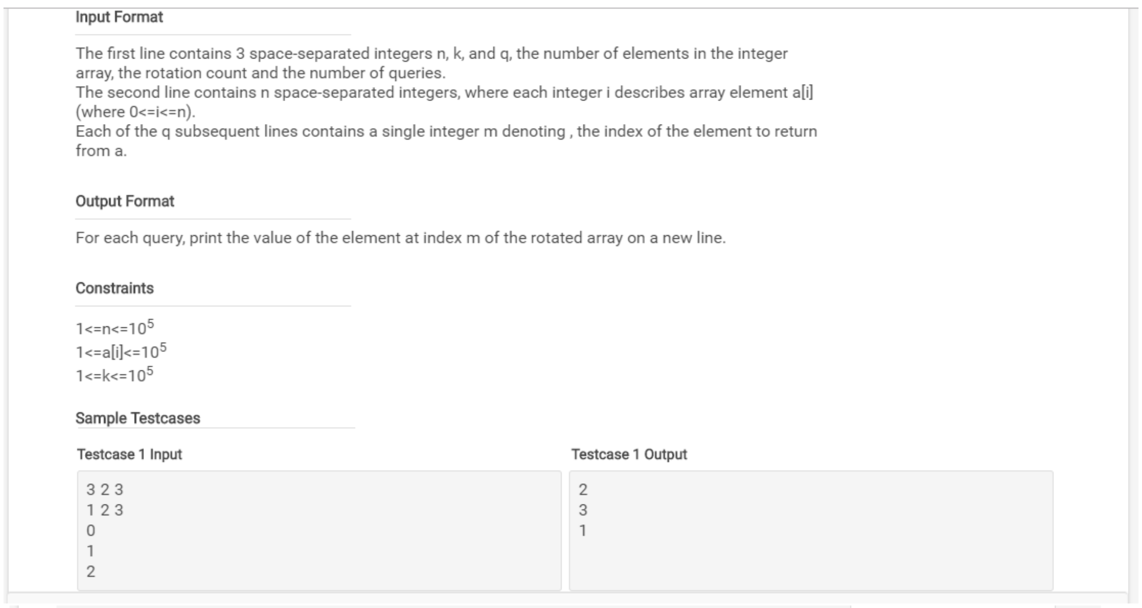
Now return the values from the zero-based indices 1 and 2 as indicated in the m array.

a[1] = 5

a[2] =3

Input Format

The first line contains 3 space-separated integers n, k, and q, the number of elements in the integer array, the rotation count and the number of queries.



Input Format

The first line contains 3 space-separated integers n, k, and q, the number of elements in the integer array, the rotation count and the number of queries.
The second line contains n space-separated integers, where each integer i describes array element a[i] (where 0<=i<=n).
Each of the q subsequent lines contains a single integer m denoting , the index of the element to return from a.

Output Format

For each query, print the value of the element at index m of the rotated array on a new line.

Constraints

$1<=n<=10^5$
$1<=a[i]<=10^5$
$1<=k<=10^5$

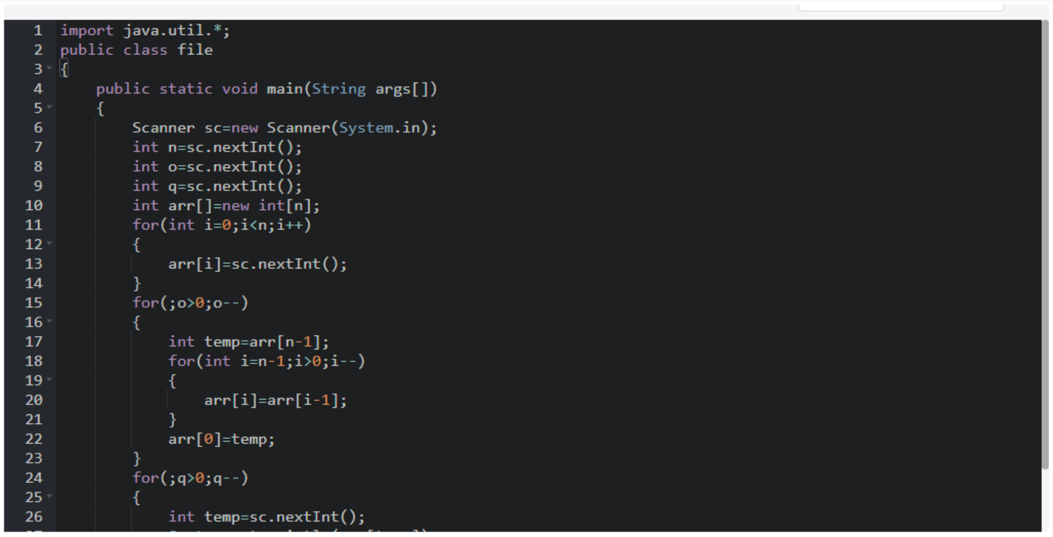
Sample Testcases

Testcase 1 Input

```
3 2 3
1 2 3
0
1
2
```

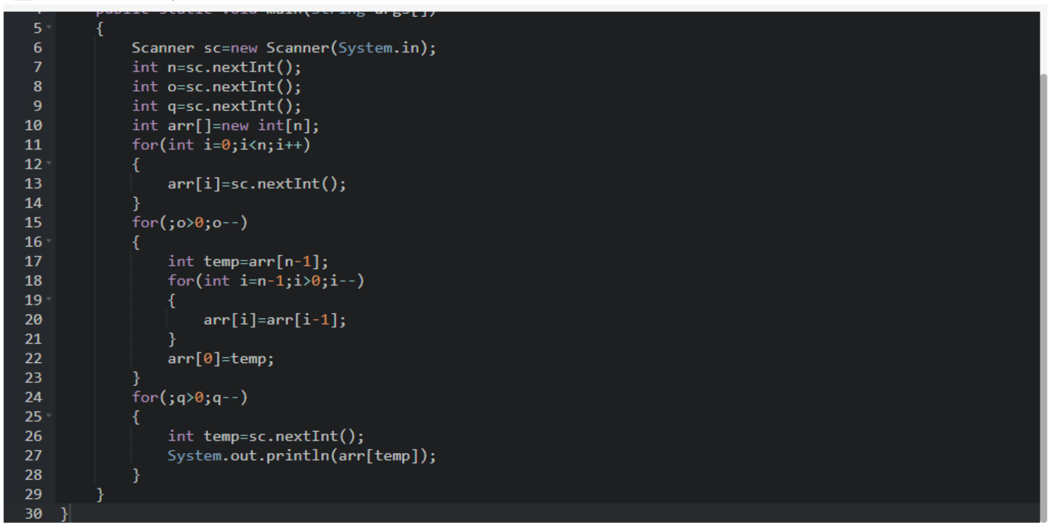
Testcase 1 Output

```
2
3
1
```

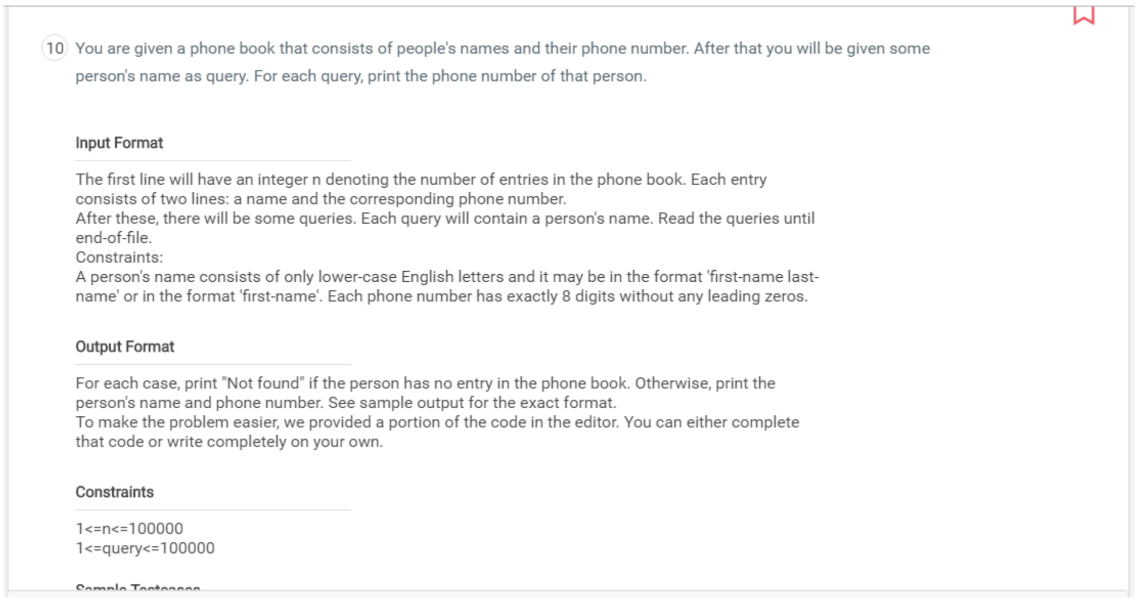


```java
import java.util.*;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int o=sc.nextInt();
        int q=sc.nextInt();
        int arr[]=new int[n];
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }
        for(;o>0;o--)
        {
            int temp=arr[n-1];
            for(int i=n-1;i>0;i--)
            {
                arr[i]=arr[i-1];
            }
            arr[0]=temp;
        }
        for(;q>0;q--)
        {
            int temp=sc.nextInt();
```

Provide custom input



```java
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int o=sc.nextInt();
        int q=sc.nextInt();
        int arr[]=new int[n];
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }
        for(;o>0;o--)
        {
            int temp=arr[n-1];
            for(int i=n-1;i>0;i--)
            {
                arr[i]=arr[i-1];
            }
            arr[0]=temp;
        }
        for(;q>0;q--)
        {
            int temp=sc.nextInt();
            System.out.println(arr[temp]);
        }
    }
}
```



10 You are given a phone book that consists of people's names and their phone number. After that you will be given some person's name as query. For each query, print the phone number of that person.

Input Format

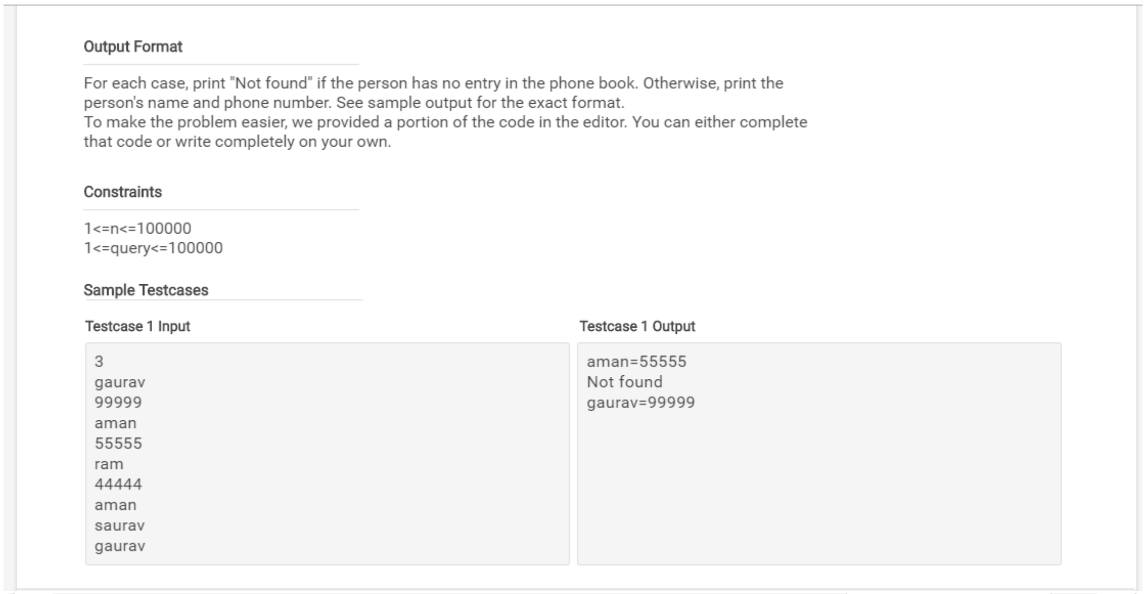
The first line will have an integer n denoting the number of entries in the phone book. Each entry consists of two lines: a name and the corresponding phone number.
After these, there will be some queries. Each query will contain a person's name. Read the queries until end-of-file.
Constraints:
A person's name consists of only lower-case English letters and it may be in the format 'first-name last-name' or in the format 'first-name'. Each phone number has exactly 8 digits without any leading zeros.

Output Format

For each case, print "Not found" if the person has no entry in the phone book. Otherwise, print the person's name and phone number. See sample output for the exact format.
To make the problem easier, we provided a portion of the code in the editor. You can either complete that code or write completely on your own.

Constraints

1<=n<=100000
1<=query<=100000



Output Format

For each case, print "Not found" if the person has no entry in the phone book. Otherwise, print the person's name and phone number. See sample output for the exact format.
To make the problem easier, we provided a portion of the code in the editor. You can either complete that code or write completely on your own.

Constraints

1<=n<=100000
1<=query<=100000

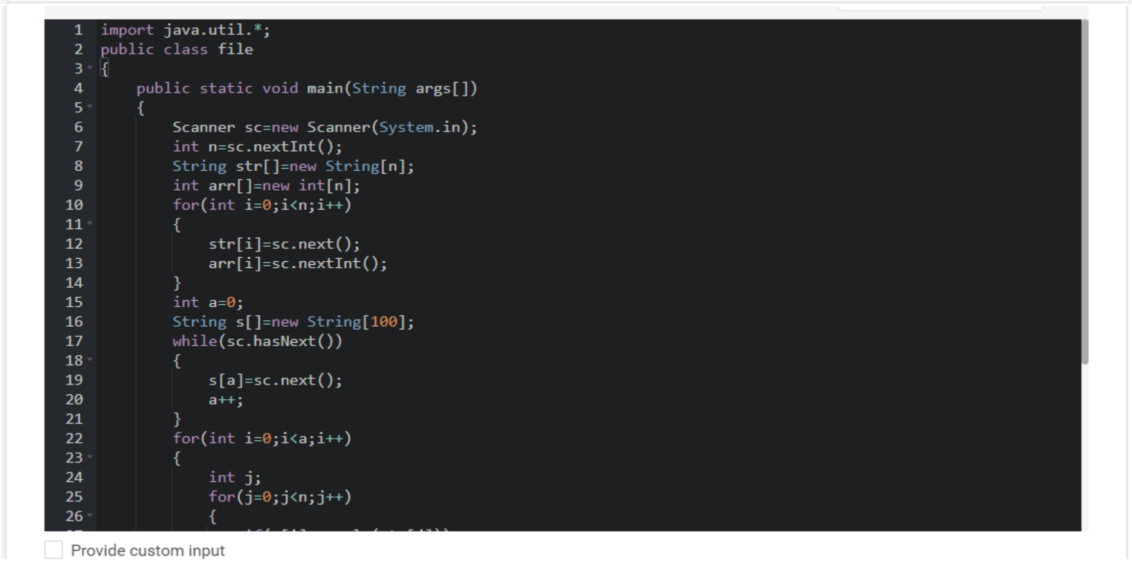
Sample Testcases

Testcase 1 Input

```
3
gaurav
99999
aman
55555
ram
44444
aman
saurav
gaurav
```

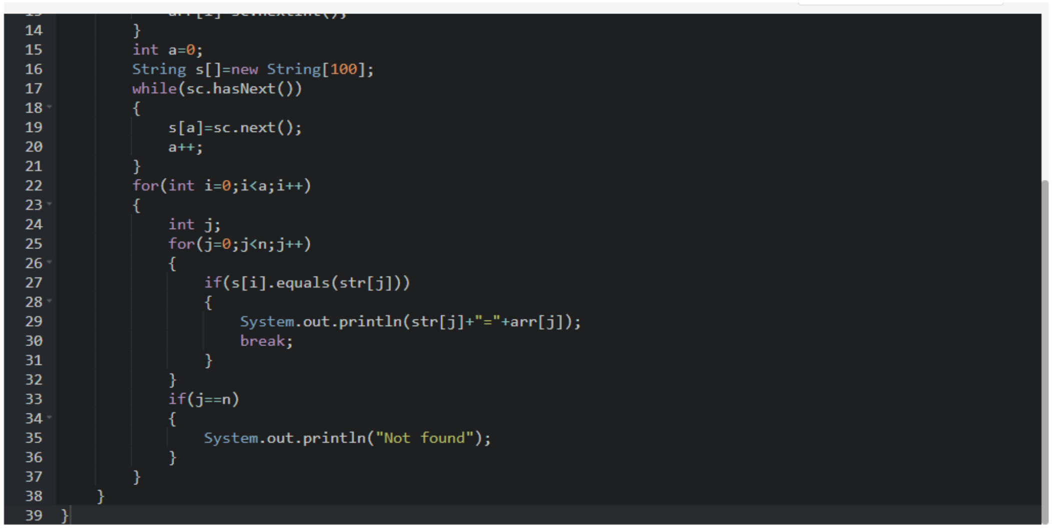
Testcase 1 Output

```
aman=55555
Not found
gaurav=99999
```



```java
import java.util.*;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        String str[]=new String[n];
        int arr[]=new int[n];
        for(int i=0;i<n;i++)
        {
            str[i]=sc.next();
            arr[i]=sc.nextInt();
        }
        int a=0;
        String s[]=new String[100];
        while(sc.hasNext())
        {
            s[a]=sc.next();
            a++;
        }
        for(int i=0;i<a;i++)
        {
            int j;
            for(j=0;j<n;j++)
            {
```

Provide custom input



```
            arr[i]=sc.nextInt();
        }
        int a=0;
        String s[]=new String[100];
        while(sc.hasNext())
        {
            s[a]=sc.next();
            a++;
        }
        for(int i=0;i<a;i++)
        {
            int j;
            for(j=0;j<n;j++)
            {
                if(s[i].equals(str[j]))
                {
                    System.out.println(str[j]+"="+arr[j]);
                    break;
                }
            }
            if(j==n)
            {
                System.out.println("Not found");
            }
        }
    }
}
```

Provide custom input

11 **TILING**

This program must calculate how many tiles are needed to tile a floor. The tiles are 8 inches by 8 inches. Tiles can be used as a whole or a part of the tile can be used. Only one usable piece can be cut from a tile. That is, if a piece is cut from a tile, the rest of the tile must be thrown away. The program accepts the length and width of the room and returns how many whole tiles are used and how many part tiles are used. The length is given in inches.

Input Format

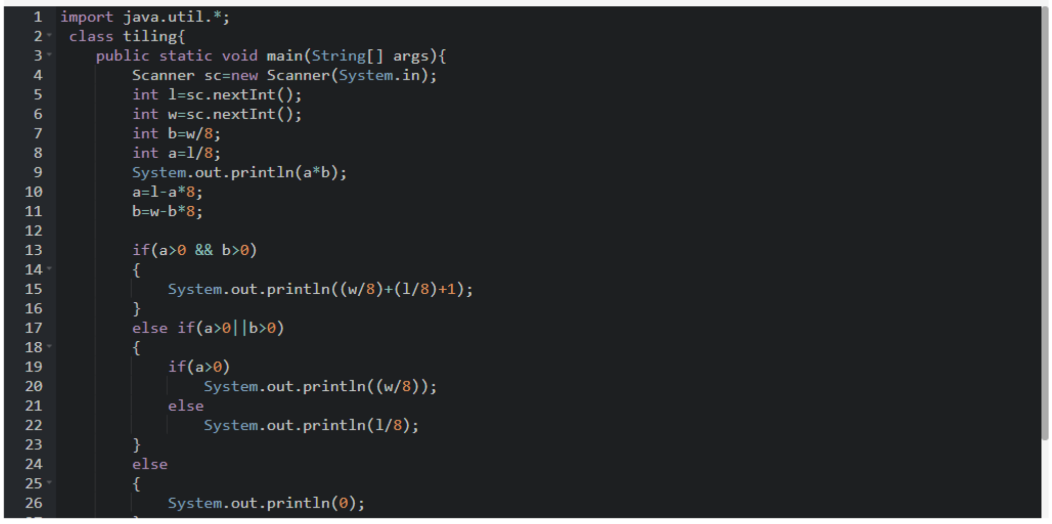
Input consists of 2 integers. The first integer corresponds to the width of the room and the second integer corresponds to the length of the room.

Output Format

Output consists of 2 integers. The first integer corresponds to the number of whole tiles used and the second integer corresponds to the number of part tiles used.

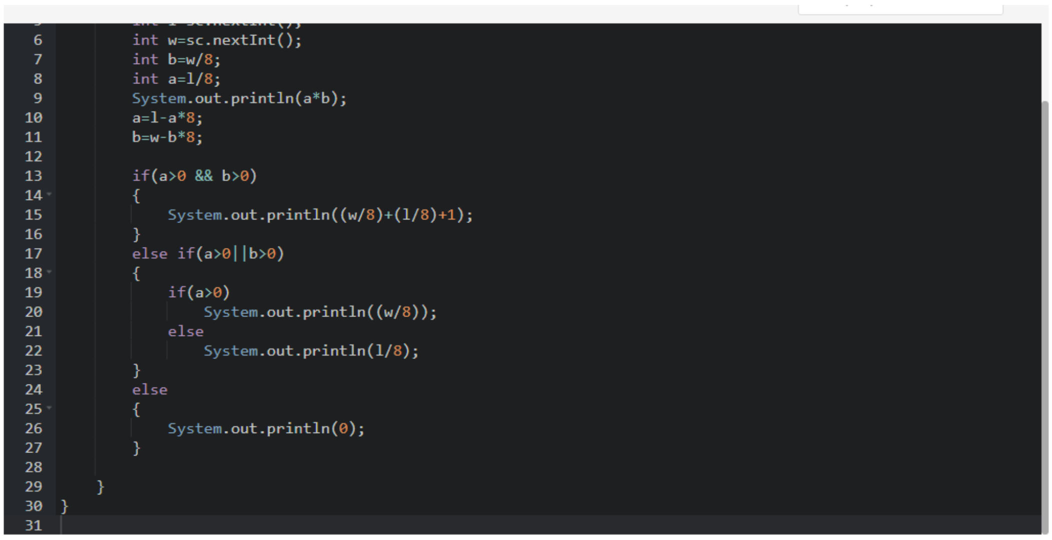
Sample Testcases

Testcase 1 Input	Testcase 1 Output
160 240	600 0



```
import java.util.*;
 class tiling{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int l=sc.nextInt();
        int w=sc.nextInt();
        int b=w/8;
        int a=l/8;
        System.out.println(a*b);
        a=l-a*8;
        b=w-b*8;

        if(a>0 && b>0)
        {
            System.out.println((w/8)+(l/8)+1);
        }
        else if(a>0||b>0)
        {
            if(a>0)
                System.out.println((w/8));
            else
                System.out.println(l/8);
        }
        else
        {
            System.out.println(0);
```

Provide custom input



```java
        int w=sc.nextInt();
        int b=w/8;
        int a=l/8;
        System.out.println(a*b);
        a=l-a*8;
        b=w-b*8;

        if(a>0 && b>0)
        {
            System.out.println((w/8)+(l/8)+1);
        }
        else if(a>0||b>0)
        {
            if(a>0)
                System.out.println((w/8));
            else
                System.out.println(l/8);
        }
        else
        {
            System.out.println(0);
        }

    }
}

```

☐ Provide custom input

12 *Lapindrome* is defined as a string which when split in the middle, gives two halves having the same characters and same frequency of each character. If there are odd number of characters in the string, we ignore the middle character and check for lapindrome. For example ***gaga*** is a lapindrome, since the two halves ***ga*** and ***ga*** have the same characters with same frequency. Also, ***abccab***, ***rotor*** and ***xyzxy*** are a few examples of lapindromes. Note that ***abbaab*** is NOT a lapindrome. The two halves contain the same characters but their frequencies do not match.
Your task is simple. Given a string, you need to tell if it is a lapindrome.

Input Format

First line of input contains a single integer **T**, the number of test cases.
Each test is a single line containing a string **S** composed of only lowercase English alphabet.

```
6
gaga
abcde
rotor
xyzxy
abbaab
ababc
```

Output Format

For each test case, output on a separate line: "YES" if the string is a lapindrome and "NO" if it is not.

YES

Output Format

For each test case, output on a separate line: "YES" if the string is a lapindrome and "NO" if it is not.

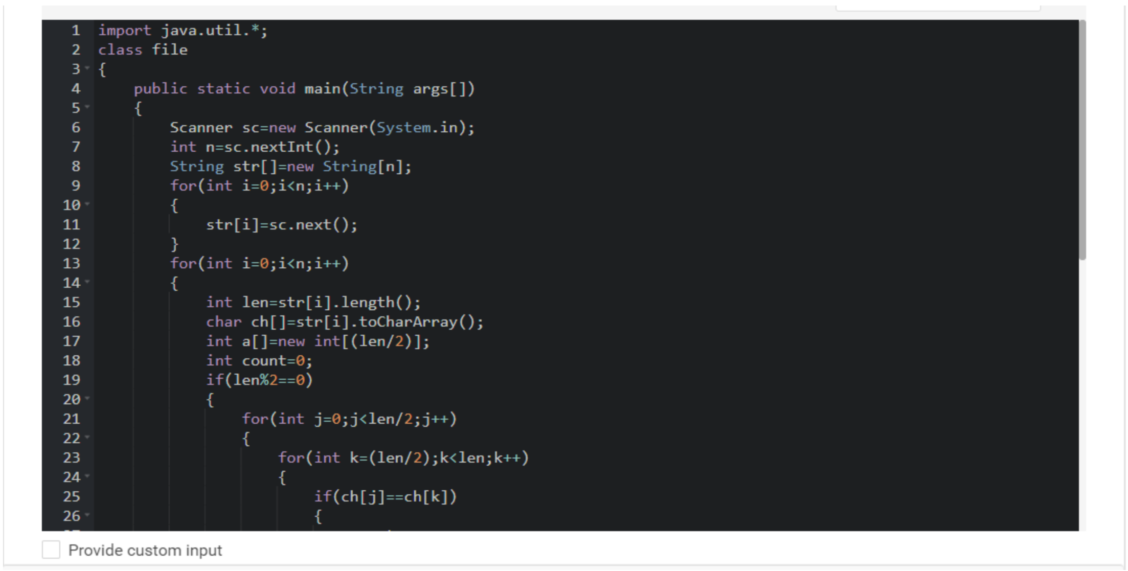
```
YES
NO
YES
YES
NO
NO
```

Constraints

1 ≤ **T** ≤ 100
2 ≤ |**S**| ≤ 1000, where |**S**| denotes the length of **S**

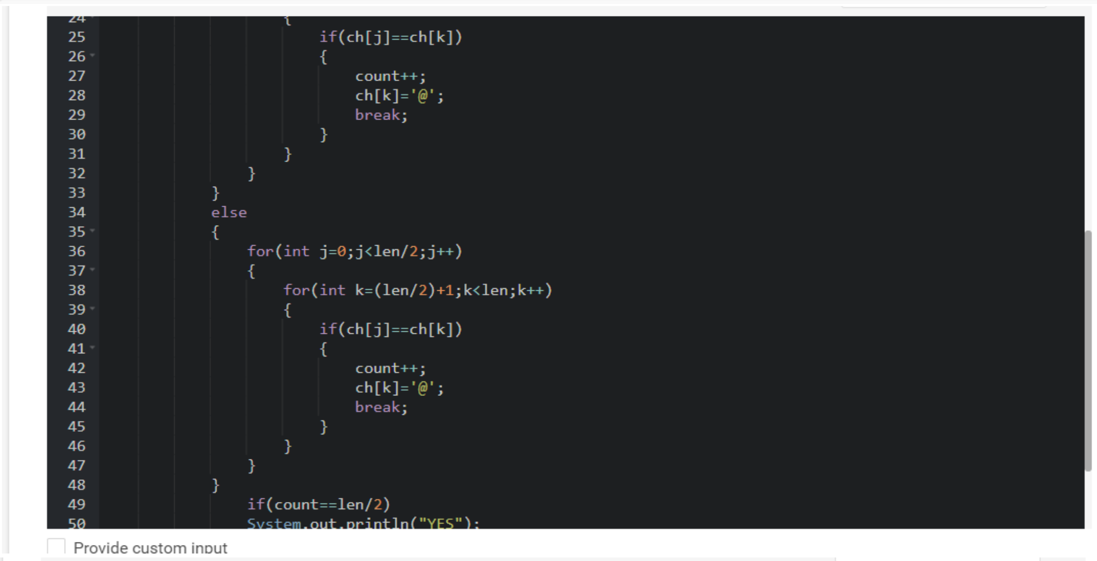
Sample Testcases

Testcase 1 Input	Testcase 1 Output
6 gaga abcde rotor xyzxy abbaab ababc	YES NO YES YES NO NO



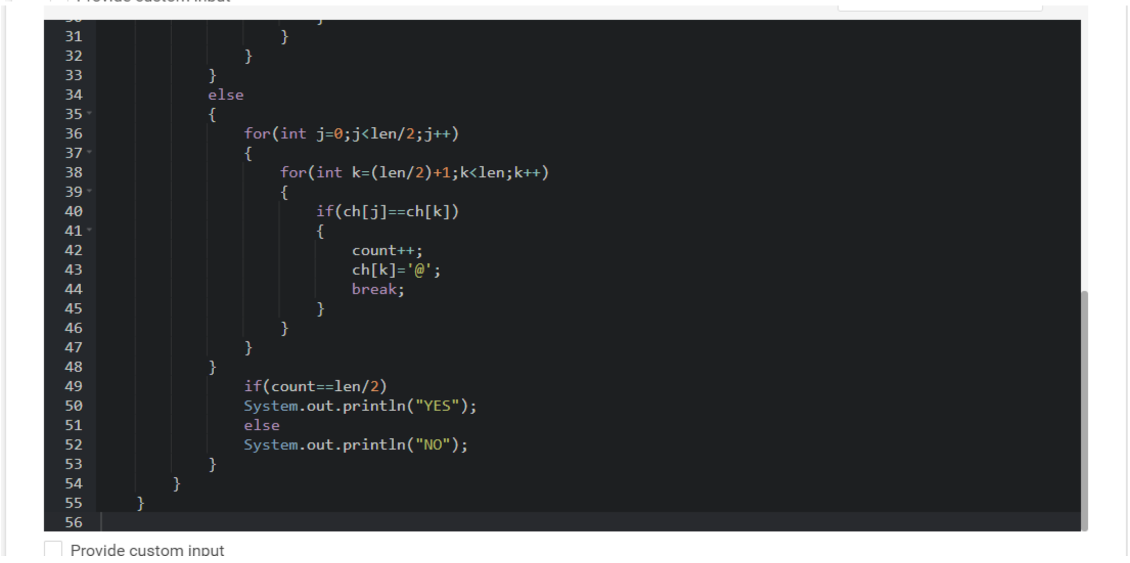
```java
import java.util.*;
class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        String str[]=new String[n];
        for(int i=0;i<n;i++)
        {
            str[i]=sc.next();
        }
        for(int i=0;i<n;i++)
        {
            int len=str[i].length();
            char ch[]=str[i].toCharArray();
            int a[]=new int[(len/2)];
            int count=0;
            if(len%2==0)
            {
                for(int j=0;j<len/2;j++)
                {
                    for(int k=(len/2);k<len;k++)
                    {
                        if(ch[j]==ch[k])
                        {
```

Provide custom input



```java
                        if(ch[j]==ch[k])
                        {
                            count++;
                            ch[k]='@';
                            break;
                        }
                    }
                }
            }
            else
            {
                for(int j=0;j<len/2;j++)
                {
                    for(int k=(len/2)+1;k<len;k++)
                    {
                        if(ch[j]==ch[k])
                        {
                            count++;
                            ch[k]='@';
                            break;
                        }
                    }
                }
            }
                if(count==len/2)
                System.out.println("YES");
```

Provide custom input



```java
                    }
                }
            }
            else
            {
                for(int j=0;j<len/2;j++)
                {
                    for(int k=(len/2)+1;k<len;k++)
                    {
                        if(ch[j]==ch[k])
                        {
                            count++;
                            ch[k]='@';
                            break;
                        }
                    }
                }
            }
                if(count==len/2)
                System.out.println("YES");
                else
                System.out.println("NO");
            }
        }
    }
```

Provide custom input

13 A Little Elephant and his friends from the Zoo of Lviv like candies very much.

There are **N** elephants in the Zoo. The elephant with number **K** (1 ≤ K ≤ N) will be happy if he receives at least A_K candies.

There are **C** candies in all in the Zoo.

The Zoo staff is interested in knowing whether it is possible to make all the **N**elephants happy by giving each elephant at least as many candies as he wants, that is, the Kth elephant should receive at least A_K candies. Each candy can be given to only one elephant. Print **Yes** if it is possible and **No** otherwise.

Input Format

The first line of the input file contains an integer **T**, the number of test cases. **T** test cases follow. Each test case consists of exactly 2 lines. The first line of each test case contains two space separated integers **N** and **C**, the total number of elephants and the total number of candies in the Zoo respectively. The second line contains **N**space separated integers $A_1, A_2, ..., A_N$.

Output Format

Output

For each test case output exactly one line containing the string **Yes** if it possible to make all elephants happy and the string **No** otherwise. Output is case sensitive. So **do not print YES or yes**.

Constraints

test case consists of exactly 2 lines. The first line of each test case contains two space separated integers **N** and **C**, the total number of elephants and the total number of candies in the Zoo respectively. The second line contains **N**space separated integers $A_1, A_2, ..., A_N$.

Output Format

Output

For each test case output exactly one line containing the string **Yes** if it possible to make all elephants happy and the string **No** otherwise. Output is case sensitive. So **do not print YES or yes**.

Constraints

1 ≤ T ≤ 1000
1 ≤ N ≤ 100
1 ≤ C ≤ 10^9
1 ≤ A_K ≤ 10000, for K = 1, 2, ..., N

Sample Testcases

Testcase 1 Input

```
2
2 3
1 1
3 7
4 2 2
```

Testcase 1 Output

```
Yes
No
```

```java
import java.util.*;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        for(int i=0;i<n;i++)
        {
            int e=sc.nextInt();
            int c=sc.nextInt();
            int sum=0;
            for(;e>0;e--)
            {
                int temp=sc.nextInt();
                sum+=temp;
            }
            if(sum>c)
            {
                System.out.println("No");
            }
            else
            {
                System.out.println("Yes");
            }
        }
```

Provide custom input

```java
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        for(int i=0;i<n;i++)
        {
            int e=sc.nextInt();
            int c=sc.nextInt();
            int sum=0;
            for(;e>0;e--)
            {
                int temp=sc.nextInt();
                sum+=temp;
            }
            if(sum>c)
            {
                System.out.println("No");
            }
            else
            {
                System.out.println("Yes");
            }
        }
    }
}
```

Provide custom input

15 Given an array of integers arr[], the task is to count all the pairs (arr[i], arr[j]) such that i + j = arr[i] + arr[j] for all 0 ≤ i < j < n.

Note: Pairs (x, y) and (y, x) are considered a single pair.

Input Format

First Line reads size of array say n
next n lines read n elements of the array

Sample Testcases

Testcase 1 Input

```
8
0
1
3
4
5
6
7
8
```

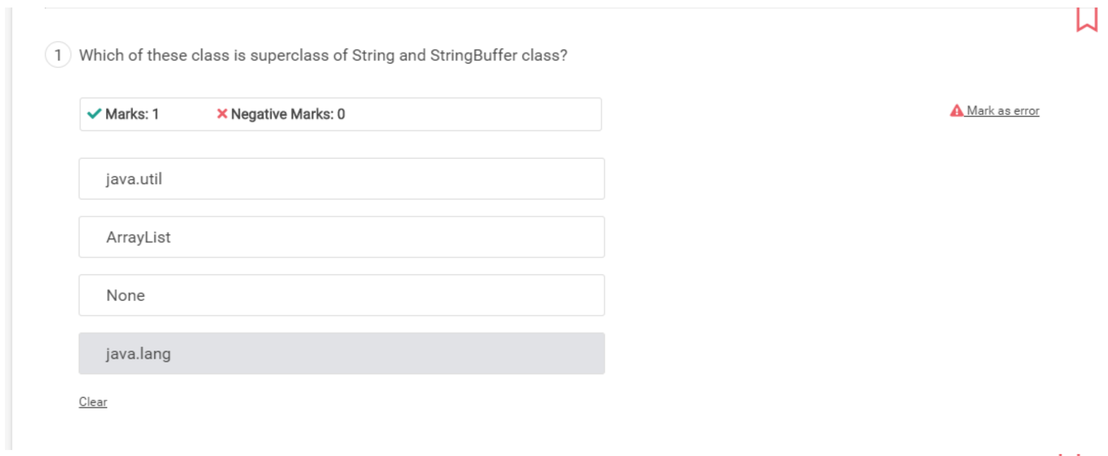
Testcase 1 Output

```
1
```

Time Limit: 0 ms **Memory Limit:** 0 kb **Code Size:** 0 kb

```java
import java.util.*;
public class file
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int arr[]=new int[n];
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }
        int count=0;
        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(arr[i]+arr[j]==i+j)
                {
                    count++;
                }
            }
        }
        System.out.println(count);
    }
}
```

Provide custom input



1 Which of these class is superclass of String and StringBuffer class?

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

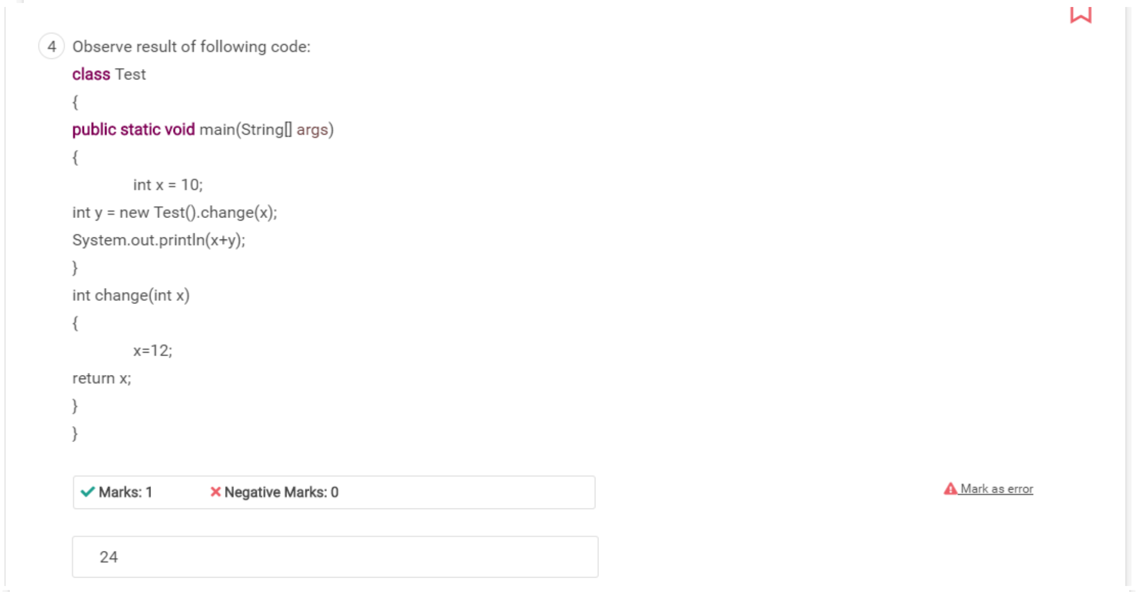
java.util

ArrayList

None

[✓ java.lang]

Clear



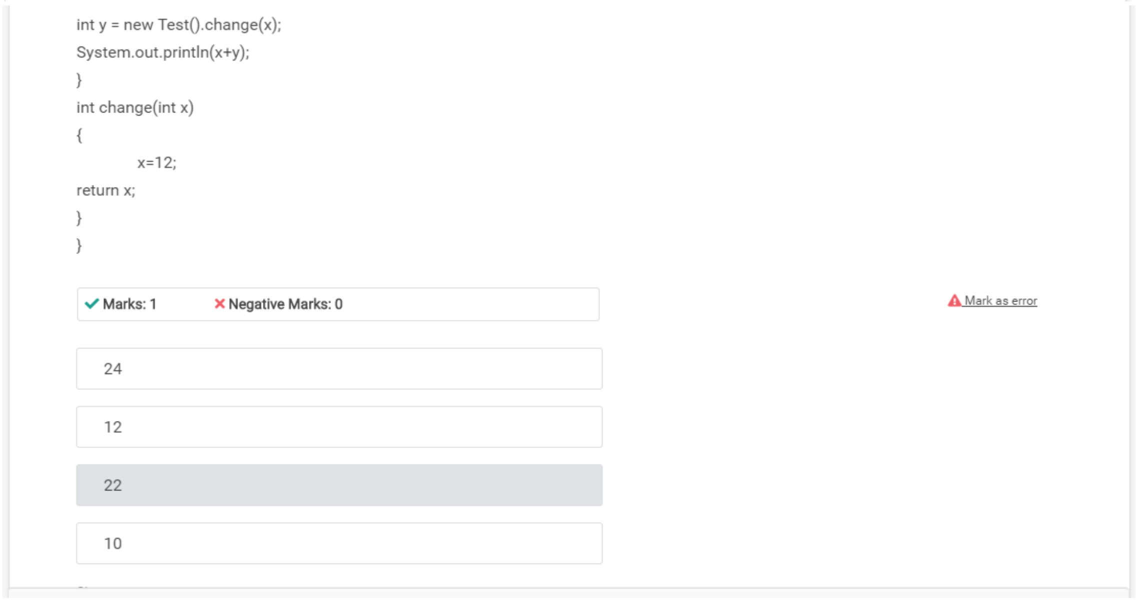
4 Observe result of following code:

```
class Test
{
public static void main(String[] args)
{
        int x = 10;
int y = new Test().change(x);
System.out.println(x+y);
}
int change(int x)
{
        x=12;
return x;
}
}
```

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

24



```
int y = new Test().change(x);
System.out.println(x+y);
}
int change(int x)
{
        x=12;
return x;
}
}
```

✓ Marks: 1 ✗ Negative Marks: 0

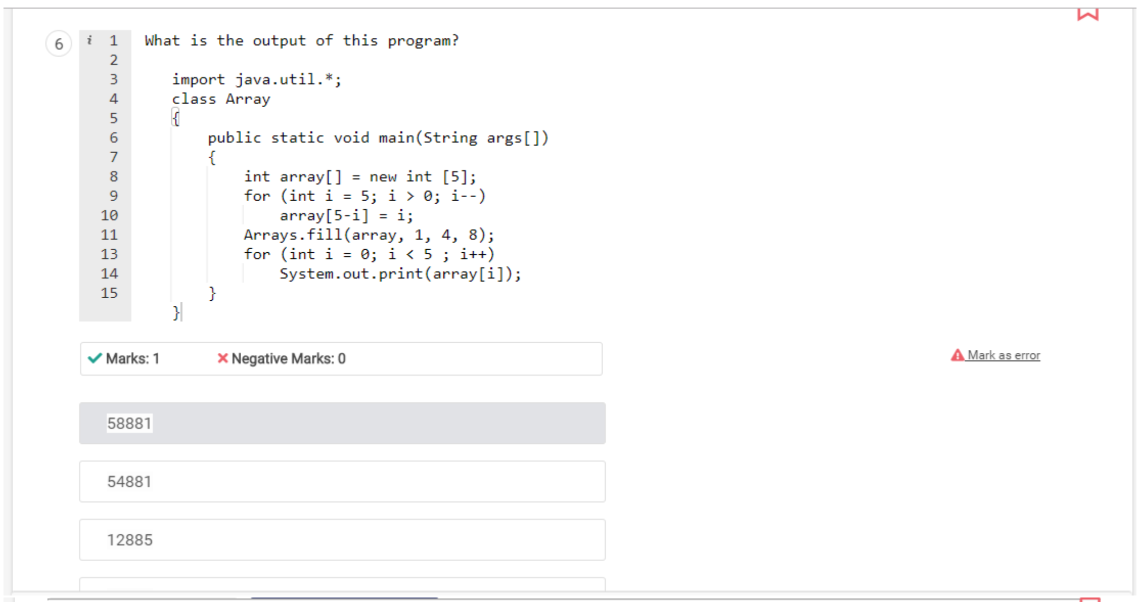
Mark as error

24

12

[✓ 22]

10



6 What is the output of this program?

```
import java.util.*;
class Array
{
    public static void main(String args[])
    {
        int array[] = new int [5];
        for (int i = 5; i > 0; i--)
            array[5-i] = i;
        Arrays.fill(array, 1, 4, 8);
        for (int i = 0; i < 5 ; i++)
            System.out.print(array[i]);
    }
}
```

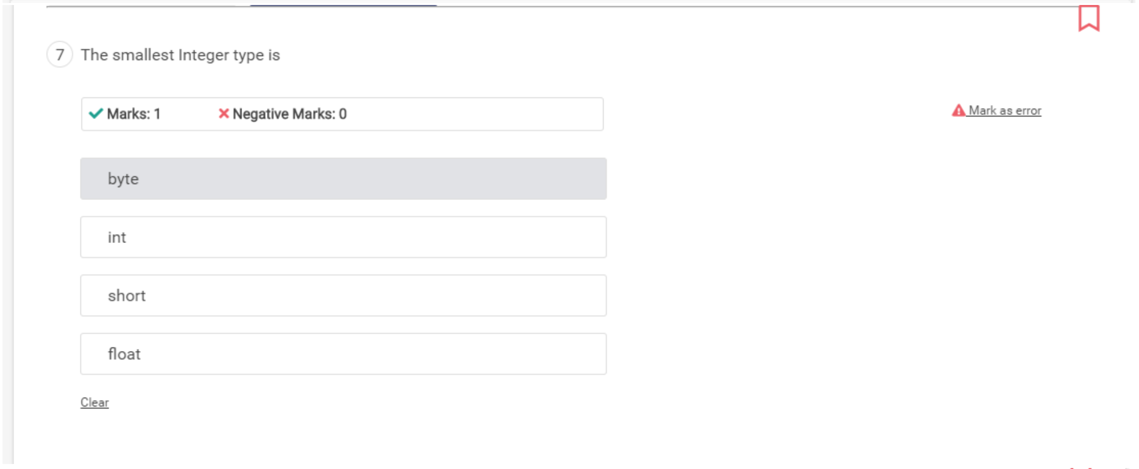
✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

[✓ 58881]

54881

12885



7 The smallest Integer type is

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

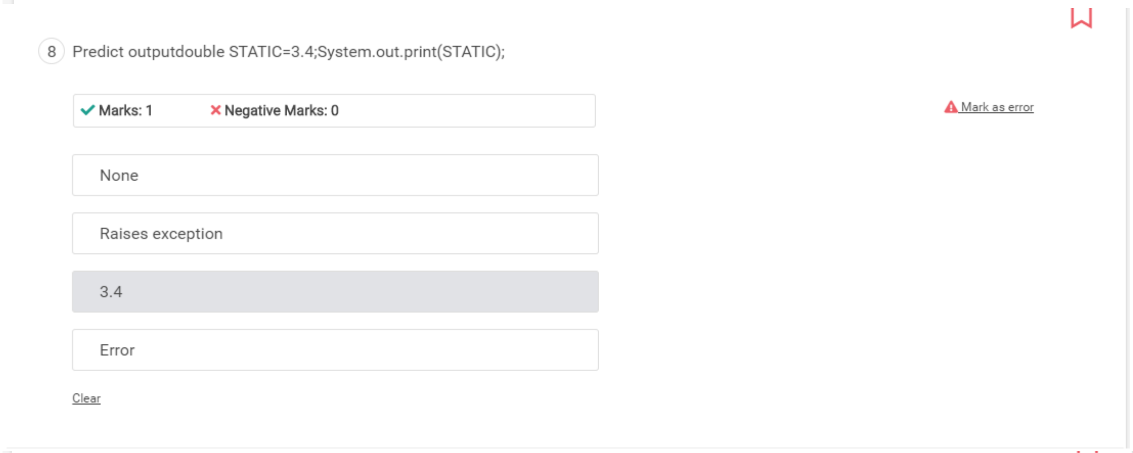
[✓ byte]

int

short

float

Clear



8 Predict outputdouble STATIC=3.4;System.out.print(STATIC);

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

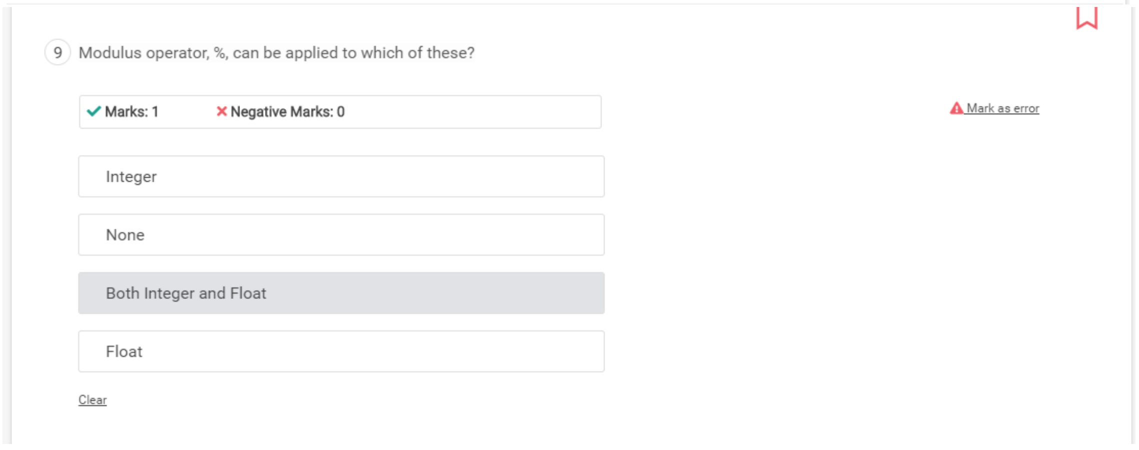
None

Raises exception

[✓ 3.4]

Error

Clear



9 Modulus operator, %, can be applied to which of these?

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

Integer

None

[✓ Both Integer and Float]

Float

Clear

10 Can 8 byte long data type be automatically type cast to 4 byte float data type?

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

False

[✓ True]

Clear

12

```
Class Test
{

public static void main(String[] args)
        {
            int[] arr[] = new int[3][];

            // Initialize the elements
            arr[0] = new int[] { 1, 2, 3};
            arr[1] = new int[] { 4, 5, 6, 7 };
            arr[2] = new int[] { 8, 9 };

            // print the array elements
            for (int[] row : arr)
                System.out.println(Arrays.toString(row));
        }
        }

```

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

Run time Error

None of These

```
            // Initialize the elements
            arr[0] = new int[] { 1, 2, 3};
            arr[1] = new int[] { 4, 5, 6, 7 };
            arr[2] = new int[] { 8, 9 };

            // print the array elements
            for (int[] row : arr)
                System.out.println(Arrays.toString(row));
        }
        }

```

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

Run time Error

None of These

Display output
[1 2 3]
[4 5 6 7]
[8 9]

[✓ Compile time Error]

14 What is Truncation is Java?

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

d) Integer value assigned to floating type

c) Floating-point value assigned to an Floating type

[✓ a) Floating-point value assigned to an integer type]

b) Integer value assigned to floating type

Clear

15 What will this code print?

```
int arr[] = new int [5];
System.out.print(arr);
```

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

c) 00000

[✓ b) value stored in arr[0].]

d) Class name@ hashcode in hexadecimal form

a) 0

16 What is the output of this program?

```
class array_output
        {
            public static void main(String args[])
            {
                int array_variable [] = new int[10];
            for (int i = 0; i < 10; ++i)
                {
                    array_variable[i] = i;
                    System.out.print(array_variable[i] + " ");
                    i++;
                }
            }
        }
```

✓ Marks: 1 ✗ Negative Marks: 0

Mark as error

[✓ a) 0 2 4 6 8]

b) 1 3 5 7 9

c) 0 1 2 3 4 5 6 7 8 9