Angry Professor:
https://www.hackerrank.com/challenges/angry-professor/problem

```
import java.util.Scanner;

public class Demo {
	static String angryProfessor(int k, int[] a) {
		int count = 0;
		for (int i = 0; i < a.length; i++) {
			if (a[i] <= 0) {
				count++;
			}
		}
		if (count >= k) {
			return "NO";
		} else {
			return "YES";
		}
	}

	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		int n = scan.nextInt();
		int k = scan.nextInt();
		int[] a = new int[n];
		for (int i = 0; i < a.length; i++) {
			a[i] = scan.nextInt();
		}
		System.out.println(angryProfessor(k, a));
	}
}
```

Migratory Birds:

Link: https://www.hackerrank.com/challenges/migratory-birds/problem

```
import java.util.Scanner;

public class Demo {
	static int migratoryBirds(int[] a) {
		int[] birds = new int[6];
		for (int i = 0; i < a.length; i++) {
			birds[a[i]]++;
		}
		int maxIndex = 0, max = -1;
		for (int i = 1; i < birds.length; i++) {
			if (birds[i] > max) {
				max = birds[i];
				maxIndex = i;
			}
		}
		return maxIndex;
	}

	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		int[] a = { 1, 1, 2, 2, 3, 4 };
		System.out.println(migratoryBirds(a));
	}
}
```

Walk In a Row:

Vanya and his friends are walking along the fence of height h and they do not want the guard to notice them. In order to achieve this the height of each of the friends should not exceed h. If the height of some person is greater than h he can bend down and then he surely won't be noticed by the guard. The height of the i-th person is equal to ai.

Consider the width of the person walking as usual to be equal to 1, while the width of the bent person is equal to 2. Friends want to talk to each other while walking, so they would like to walk in a single row. What is the minimum width of the road, such that friends can walk in a row and remain unattended by the guard?

Input Format

The first line of the input contains two integers n and k (1 ≤ n ≤ 1000, 1 ≤ k ≤ 1000) — the number of friends and the height of the fence, respectively.

The second line contains n integers ai (1 ≤ ai ≤ 2h), the i-th of them is equal to the height of the i-th person.

Output Format

Print a single integer — the minimum possible valid width of the road.

Input:

5 6

4 5 14 7 5

Output:

7

Solution:

```
import java.util.Scanner;
public class Demo {
     static int walkWidth(int[] a, int k) {
          int width = 0;
          for (int i = 0; i < a.length; i++) {
               if (a[i] <= k) {
                    width = width + 1;
               } else {
                    width = width + 2;
               }
          }
          return width;
     }

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

```
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        int k = scan.nextInt();
        int[] a = new int[n];
        for (int i = 0; i < a.length; i++) {
            a[i] = scan.nextInt();
        }
        System.out.println(walkWidth(a, k));
    }
}
```

Circular Array Rotation

Link:

https://www.hackerrank.com/challenges/circular-array-rotation/problem

Solution:

```
import java.util.HashMap;
import java.util.Map.Entry;
import java.util.Scanner;
public class Test {
	static int[] circularArrayRotation(int[] a, int k, int[] q) {
		int[] res = new int[q.length];
		int[] b = new int[a.length];
		for (int i = 0; i < a.length; i++) {
			b[(i + k) % a.length] = a[i];
		}
		for (int i = 0; i < q.length; i++) {
			res[i] = b[q[i]];
		}
		return res;
	}
	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		int n = scan.nextInt();
		int k = scan.nextInt();
		int q = scan.nextInt();
		int a[] = new int[n];
		int queries[] = new int[q];
		for (int i = 0; i < a.length; i++) {
			a[i] = scan.nextInt();
		}
		for (int i = 0; i < queries.length; i++) {
			queries[i] = scan.nextInt();
		}
		int[] res = circularArrayRotation(a, k, queries);
		for (int i = 0; i < res.length; i++) {
			System.out.println(res[i]);
		}
	}
}
```

Merge Sorted Array:

```
import java.util.Scanner;

public class Test {
	static int[] mergeSortedArray(int[] ar1, int[] ar2) {
		int i = 0, j = 0, k = 0;
		int[] res = new int[ar1.length + ar2.length];
		while (i < ar1.length && j < ar2.length) {
			if (ar1[i] < ar2[j]) {
				res[k] = ar1[i];
				i++;
				k++;
			} else {
				res[k] = ar2[j];
				j++;
				k++;
			}
		}
		while (i < ar1.length) {
			res[k] = ar1[i];
			i++;
			k++;
		}
		while (j < ar2.length) {
			res[k] = ar2[j];
			j++;
			k++;
		}
		return res;
	}

	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		int[] ar1 = { 3, 5, 9, 12, 15 };
		int[] ar2 = { 1, 6, 8 };
		int[] res = mergeSortedArray(ar1, ar2);
		for (int i = 0; i < res.length; i++) {
			System.out.print(res[i] + " ");
		}
	}
}
```

AES NUMBER:

Write a program to print the number of aes numbers in the given range.

NOTE: AES number is the number which contains exactly 4 divisors then it is called as aes number

```
package mock;

import java.util.Scanner;
 class Demo{

      static boolean isAesNumber(int n) {
                int count = 0;
                for (int i = 1; i <=n; i++) {
                      if(n%i==0) {
                            count++;
                      }
                }

                if(count==4) {
                      return true;
                }
                else {
                      return false;
                }
          }

     public static void main(String[] args)
     {
          Scanner scan = new Scanner(System.in);
          int l = scan.nextInt();
          int r = scan.nextInt();
          int aes = 0;

          for(int i=l; i<=r; i++)
          {
                if(isAesNumber(i) == true) {
                      aes++;
                }
          }
```

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

Devu's friendship test

Devu has **n** weird friends. Its his birthday today, so they thought that this is the best occasion for testing their friendship with him. They put up conditions before Devu that they will break the friendship unless he gives them a grand party on their chosen day. Formally, **i**th friend will break his friendship if he does not receive a grand party on **di**th day.

Devu despite being as rich as Gatsby, is quite frugal and can give at most one grand party daily. Also, he wants to invite only one person in a party. So he just wonders what is the maximum number of friendships he can save. Please help Devu in this tough task !!

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.
- First line will contain a single integer denoting **n**.
- Second line will contain **n** space separated integers where **i**th integer corresponds to the day **di**th as given in the problem.

Output

Print a single line corresponding to the answer of the problem.

Sample Input:

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

Sample Output:

```
2
1
```

```
import java.util.Scanner;
```

```
class Sorting{
	static void selectionSort(int[] a) {
		for (int i = 0; i < a.length - 1; i++) {
			int min_i = i;
			for (int j = i + 1; j < a.length; j++) {
				if (a[j] > a[min_i]) {
					min_i = j;
				}
			}
			int temp = a[i];
			a[i] = a[min_i];
			a[min_i] = temp;
		}
	}
}
 class Demo{

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

		for (int i = 0; i < d.length; i++) {
			d[i] = scan.nextInt();
		}

		Sorting.selectionSort(d);

		int count = 0;
		for (int i = 0; i < d.length; i++) {
			if(d[i] != d[i+1]) {
				count++;
			}
		}

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

Linear search

```
class Searching{
       public static int linearSearch(int[] a, int key) {
             for (int i = 0; i < a.length; i++) {
                  if(a[i] == key) {
                        return i;
                  }
             }
              return -1;
       }

      public static void main(String[] args)
      {
            int[] a = {50,10,33,40,26};
            int key = 99;
            System.out.println(linearSearch(a, key));
      }

}
```

Binary search

```java
import java.util.Arrays;

class tr {
	public static int binarySearch(int[] a, int key) {
		int l = 0,h = a.length-1, mid = 0;

		while(l <= h) {
			mid = (l+h)/2;
			if(key == a[mid]) {
				return mid;
			}
			else if(key < a[mid]) {
				h = mid-1;
			}
			else {
				l = mid-1;
			}
		}
		return -1;
	 }

	public static void main(String[] args)
	{
		int[] a = {3,5,6,8,12,15,16,19,21};
		Arrays.sort(a);
		int key = 15;
		System.out.println(binarySearch(a, key));
	}

}
```

Bubble Sort

```
public class Sorting {
	static void bubbleSort(int[] a) {
		for (int i = 0; i < a.length - 1; i++) {
			for (int j = 0; j < a.length - i - 1; j++) {
				if (a[j] < a[j + 1]) {
					int temp = a[j];
					a[j] = a[j + 1];
					a[j + 1] = temp;
				}
			}
		}
	}

	public static void main(String[] args) {
		int[] a = { 3, 2, 7, 5, 9 };
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}
		System.out.println();
		bubbleSort(a);
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}
	}
}
```

Selection Sort

```
public class Sorting {
	static void selectionSort(int[] a) {
		for (int i = 0; i < a.length - 1; i++) {
			int min_i = i;
			for (int j = i + 1; j < a.length; j++) {
				if (a[j] > a[min_i]) {
					min_i = j;
				}
			}
			int temp = a[i];
			a[i] = a[min_i];
			a[min_i] = temp;
		}
	}

	public static void main(String[] args) {
		int[] a = { 3, 2, 7, 5, 9 };
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}
		System.out.println();
		selectionSort(a);
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}
	}
}
```

Insertion sort

```
public class Sorting {
	static void insertionSort(int[] a) {
		int j = 0;
		for (int i = 1; i < a.length; i++) {
			j = i - 1;
			int key = a[i];
			while (j >= 0 && a[j] < key) {
				a[j + 1] = a[j];
				j--;
			}
			a[j + 1] = key;
		}
	}

	public static void main(String[] args) {

		int[] a = { 3, 2, 7, 5, 9 };
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}
		System.out.println();
		insertionSort(a);
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}
	}
}
```