import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.Arrays;  
  
  
public class Main {

// Complessità: O(nlogn)  
static int count(int arr[], int low, int high, int x) {  
 int count;  
 // Impera: Se l'array non è valido oppure l'elemento è diverso da x  
 if ((low > high) || (low == high && arr[low] != x))  
 return 0;  
 // Impera: Se l'el è uguale a x, lo conto  
 if (low == high && arr[low] == x)  
 return 1;  
 // Dividi: conta le occorrenze nella prima metà e nella seconda metà dell'array  
 count = *count*(arr, low, (low + high) / 2, x) + *count*(arr, 1 + (low + high) / 2, high, x);  
 return count;  
}

public static void main(String[] args) throws IOException {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 String line = reader.readLine();  
 String[] tokens = line.split(" ");  
 int ntest = Integer.*parseInt*(tokens[0]);  
 while (ntest-- > 0) {  
 tokens = reader.readLine().split(" ");  
 int x = Integer.*parseInt*(tokens[0]);  
 int n = Integer.*parseInt*(tokens[1]);  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = Integer.*parseInt*(tokens[i+2]);  
 }  
 System.*out*.println(*count*(arr, 0, n - 1, x));  
 }  
 }  
}

PROBLEMA 2

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
  
public class Main {  
  
 static ArrayList<Integer> *primes* = new ArrayList<Integer>();  
 static ArrayList<Integer> *ans* = new ArrayList<Integer>();  
  
 // Verifica se è numero primo  
 static boolean isPrime(int x) {  
 int sqroot = (int) Math.*sqrt*(x);  
 // Caso n = 1  
 if (x == 1)  
 return false;  
 // Vede se è divisibile  
 for (int i = 2; i <= sqroot; i++)  
 if (x % i == 0)  
 return false;  
 // Altrimenti è primo  
 return true;  
 }  
  
 static void print() {  
 int length = *ans*.size();  
 for (int i = 0; i < length; i++)  
 System.*out*.print(*ans*.get(i) + " ");  
 System.*out*.println();  
 }  
  
 // Funzione di backtrack: Calcola tutti i numeri primi la cui somma è S. Complessità O(2^n)  
 static void primeSum(int total, int N, int S, int index) {  
 // se ho raggiunto la fine  
 if (total == S && *ans*.size() == N) {  
 *print*();  
 return;  
 }  
 // se ho raggiunto il massimo  
 if (total > S || index == *primes*.size() || *ans*.size() >= N)  
 return;  
 // Set: Aggiungi il numero al set ans  
 *ans*.add(*primes*.get(index));  
 *primeSum*(total + *primes*.get(index), N, S, index + 1);  
 // backtrack: rimuovi el dal set  
 *ans*.remove(*ans*.size() - 1);  
 *primeSum*(total, N, S, index + 1);  
 }  
  
 // Genero tutti i num primi  
 static void allPrime(int N, int S, int P) {  
 for (int i = P + 1; i <= S; i++) {  
 // se i è primo, lo aggiungo all'array di num primi  
 if (*isPrime*(i))  
 *primes*.add(i);  
 }  
 if (*primes*.size() < N)  
 return;  
 *primeSum*(0, N, S, 0);  
 }  
  
  
 public static void main(String[] args) throws IOException {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 String line = reader.readLine();  
 String[] tokens = line.split(" ");  
 int ntest = Integer.*parseInt*(tokens[0]);  
 int num = 0;  
 while (ntest-- > 0) {  
 tokens = reader.readLine().split(" ");  
 int S = Integer.*parseInt*(tokens[0]);  
 int N = Integer.*parseInt*(tokens[1]);  
 int P = Integer.*parseInt*(tokens[2]);  
 System.*out*.printf("CASO DI TEST %d\n", ++num);  
 *allPrime*(N, S, P);  
 }  
}

}