```
//AnandVinod
//Junior-Patolli

import java.util.*;
import java.io.*;

public class AnandV_JRC4Patolli {

    public static void main(String[] args) throws IOException {
        Scanner scan = new Scanner(new File("C:\\Users\\Anand Vinod\\Desktop\\data.txt"));
        while(scan.hasNextLine()) {
            String []temp = scan.nextLine().split(" ");
            ArrayList<Integer> locations = new ArrayList<>();
            locations.add(Integer.valueOf(temp[0]));
            locations.add(Integer.valueOf(temp[1]));
            locations.add(Integer.valueOf(temp[2]));
            int currLocation = Integer.parseInt(temp[3]);
            ArrayList<Integer> rolls = new ArrayList<>();
            for(int i = 0; i < Integer.parseInt(temp[4]); i++)
            {
                rolls.add(Integer.parseInt(temp[i+5])) ;
            }
            for(int roll : rolls)
            {
                currLocation = move(locations, currLocation, roll);
            }
            if(currLocation == 52)
                System.out.println("GAME OVER");
            else
                System.out.println(currLocation);
        }
    }

    private static int move(ArrayList<Integer> locations, int currLocation, int roll) {
        if(locations.contains(roll+currLocation) || currLocation + roll >52 || currLocation ==
        52)
            return currLocation;
        else if(isPrime(currLocation + roll))
        {
            currLocation += roll;
            int triedRet = 6;
            for(int location : locations)
            {
                if(triedRet + currLocation >= location && currLocation < location)
                {
                    triedRet = location - currLocation -1;
                }
            }
            return currLocation + triedRet;
        }
        else if(isPerfectSquare(currLocation + roll))
        {
            currLocation += roll;
            int triedRet = 6;
            for(int location : locations)
            {
                if(currLocation - triedRet <= location && currLocation > location)
                {
                    triedRet = currLocation - location - 1;
                }
            }
            return currLocation - triedRet;
        }
        if(willMoveHorzAndVert(currLocation, roll))
        {
            int currentMove = roll;
            while((currLocation + currentMove) % roll != 0 && currentMove > 0)
            {
                if(locations.contains(currLocation + currentMove -1) && currentMove > 1 &&
```

```
                (currLocation+currentMove-1)%roll == 0)
                {
                    currentMove -= 2;
                }
                else
                    currentMove--;
            }
            if(!locations.contains(currLocation + currentMove))
                return currLocation + currentMove;
            else
                return currLocation;
        }
        return currLocation + roll;
    }
    public static boolean willMoveHorzAndVert(int currPos, int roll)
    {
        ArrayList<int []> possibilities = new ArrayList<>();
        for(int i = 0 ; i < 5; i ++)
        {
            possibilities.add(new int[] {6 + 5 * i, 8 + 5 * i});
        }

        for(int i = 0 ; i < 4; i++)
        {
            possibilities.add(new int [] {34 + 5 * i, 36 + 5 * i});
        }

        for(int [] range : possibilities)
        {
            if(currPos <= range[0] && currPos + roll >= range[1])
                return true;
        }

        return false;
    }
    public static boolean isPrime(int a){

        for(int i = 2; i <= a/2; i++)
        {
            if(a % i == 0)
                return false;
        }
        return !(a==1);
    }

    public static boolean isPerfectSquare(int a)
    {
        return Math.sqrt(a) == (int)Math.sqrt(a) && a > 4;
    }

}
```