COMPUTER SOCIETY OF INDIA (CSI)
COEP TECH STUDENT CHAPTER

PRESENTS

CODE QUEST 5.0

EDITORIAL

BOLTBOT AND TALLYTRON

PROBLEM STATEMENT

The COEP Robotics Team is making a new robot "BoltBot" for their upcoming Robocon Competition. This year's Robocon theme is about making a Robot which has to destroy targets arranged in a circle by moving along the circle. The team wants to count the number of revolutions BoltBot makes along the circle for accomplishing this goal. BoltBot is currently in the developing stage and is not fully ready for the competition.
There is a revolutions counting robot "TallyTron" which is being used to count the revolutions BoltBot makes around the circle. TallyTron is placed at the starting line of the circular track and starts the count from 0. Every time BoltBot arrives at the starting line moving in the same direction as the last time it departed from the starting line, TallyTron increases the number of revolutions that BoltBot has completed by 1. If BoltBot crosses the starting line or changes direction at the starting line, TallyTron considers

PROBLEM STATEMENT

the new direction as the direction BoltBot last touched the starting line. TallyTron only remembers the last direction in which BoltBot touched the starting line. During a revolution, BoltBot can change directions any number of times, but as long as it eventually touches the starting line in the same direction as she it touched it, the count of revolutions in TallyTron increases by 1.

The Robotics team have recently recruited some freshers in their team and have asked them to use TallyTron to count the number of revolutions BoltBot makes without meddling with BoltBot's program as it will be used for Robocon competition. However, Cameron who is one of the recruits is keen on understanding new concepts and so decides to secretly check out BoltBot's working program. While inspecting the code, he accidently makes some changes in the program

PROBLEM STATEMENT

and is unable to recover the program (as the changes in the software cannot be undone) !! This has resulted in some undesirable changes in the robot's working.

Due to the changes, BoltBot cannot move continuously. It moves some distance along the circle, then stops to recharge its battery. However, after it has recharged the battery it cannot remember which direction it was moving in previously. So it randomly picks one of the directions (Clockwise of anti-clockwise) and starts moving in that direction from the same position it had stopped.

BoltBot begins at the starting line and is initially facing in the direction of its first move. It runs a total of N times, taking breaks in between. Given the information of the distance Di units

PROBLEM STATEMENT

BoltBot has run, and the direction Ai it has taken, clockwise or anticlockwise when it ran the i-th time (1 for clockwise, -1 for anitclockwise), for all i from 1,...,N, can you tell the number of revolutions that would be reported by TallyTron at the end?

INPUT FORMAT

The first line of the input gives the number of test cases, T. T test cases follow.
The first line of each test case contains two positive integers L and N, the length of the circular track in units, and the number of times BoltBot has moved respectively.
The next N lines describe BoltBot's movess. The i-th line contains two integers Di, the distance in units BoltBot has moved and Ai, the direction it has taken (clockwise or anticlockwise) respectively during the i-th move. Ai will always be either '1' (denoting clockwise direction) or '-1' (denoting anticlockwise direction).

OUTPUT FORMAT & CONSTRAINTS

For each test case, output a single integer which represents the number of revolutions counted by TallyTron.

1<=L<=10^9

1<=N<=10^4

1<=Di<=10^9

SAMPLE TESTCASES

SAMPLE INPUT:

```
2
5 3
8 C
3 C
6 C
8 4
5 C
9 C
8 C
20 C
```

SAMPLE OUTPUT:

```
3
5
```

SOLUTION

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

public class Solution {
    public static long laps(int[] dist, int[] dir, int l, int n){
        long rem=0;
        long curr=-1;
        long laps=0;
        for (int i=0;i<n;i++){
            if (i==0){
                if (dir[i]==-1){
                    curr=-1;
                }
                else{
                    curr=1;
                }
                laps += (rem+dist[i])/l;
                rem = (rem+dist[i])%l;
            }
            else{
                if (dir[i]==curr){
                    laps += (rem+dist[i])/l;
                    rem = (rem+dist[i])%l;
                }
                else{
                    laps += (Math.abs(dist[i]-rem))/l;
                    rem = (Math.abs(dist[i]-rem))%l;
                    curr=dir[i];
                }
            }
        }
        return laps;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

```

SOLUTION

```
38          int T = scanner.nextInt();
39
40          for (int t = 0; t < T; t++) {
41              int L = scanner.nextInt();
42              int N = scanner.nextInt();
43
44              int[] distances = new int[N];
45              int[] directions = new int[N];
46
47              for (int i = 0; i < N; i++) {
48                  distances[i] = scanner.nextInt();
49                  directions[i] = scanner.nextInt();
50              }
51              System.out.println(laps(distances,directions,L,N));
52          }
53          scanner.close();
54      }
55  }
```