AcWing-4740：跑圈

阿达正在一个长度为 L 的环形跑道上练习跑步。

为了更专注于跑步，阿达专门准备了一台机器来统计她跑的圈数。

机器放置在跑道的起跑线上，从 0 开始计数。

每当阿达离开起跑线时（直接越过起跑线或在起跑线位置处改变方向并离开起跑线），她的面朝方向就会被机器记录。

机器只会实时记录她最近一次离开起跑线时的面朝方向。

每当阿达到达起跑线位置时，只要其面朝方向与机器记录的上次离开起跑线时的面朝方向相同，机器计数就会加 1。

阿达从起跑线处开始跑步。

她的耐力有限，无法将计划的训练量一口气完成。

因此，每跑一段距离，她都会原地休息一段时间，用来恢复体力。

不幸的是，阿达的记忆力并不是很好，每当她休息完再次开始跑步时，她都会忘了之前面朝的方向。

这时，她只能随意选择一个方向（顺时针或逆时针），并面朝该方向从她停下的位置开始继续跑步。

具体的说，她一共进行了 N 段跑步，其中第 i 段跑步的距离为 D_i，跑步时的面朝方向为 C_i。

请你计算，在阿达完成跑步后，机器最终记录的圈数。

输入格式

第一行包含整数 T，表示共有 T 组测试数据。

每组数据第一行包含两个整数 L, N。

接下来 N 行，每行包含一个整数 D_i 和一个字符 C_i，分别表示阿达一段跑步的距离和面朝方向。C_i 只可能是 `C`（表示顺时针方向）或 `A`（表示逆时针方向）。

输出格式

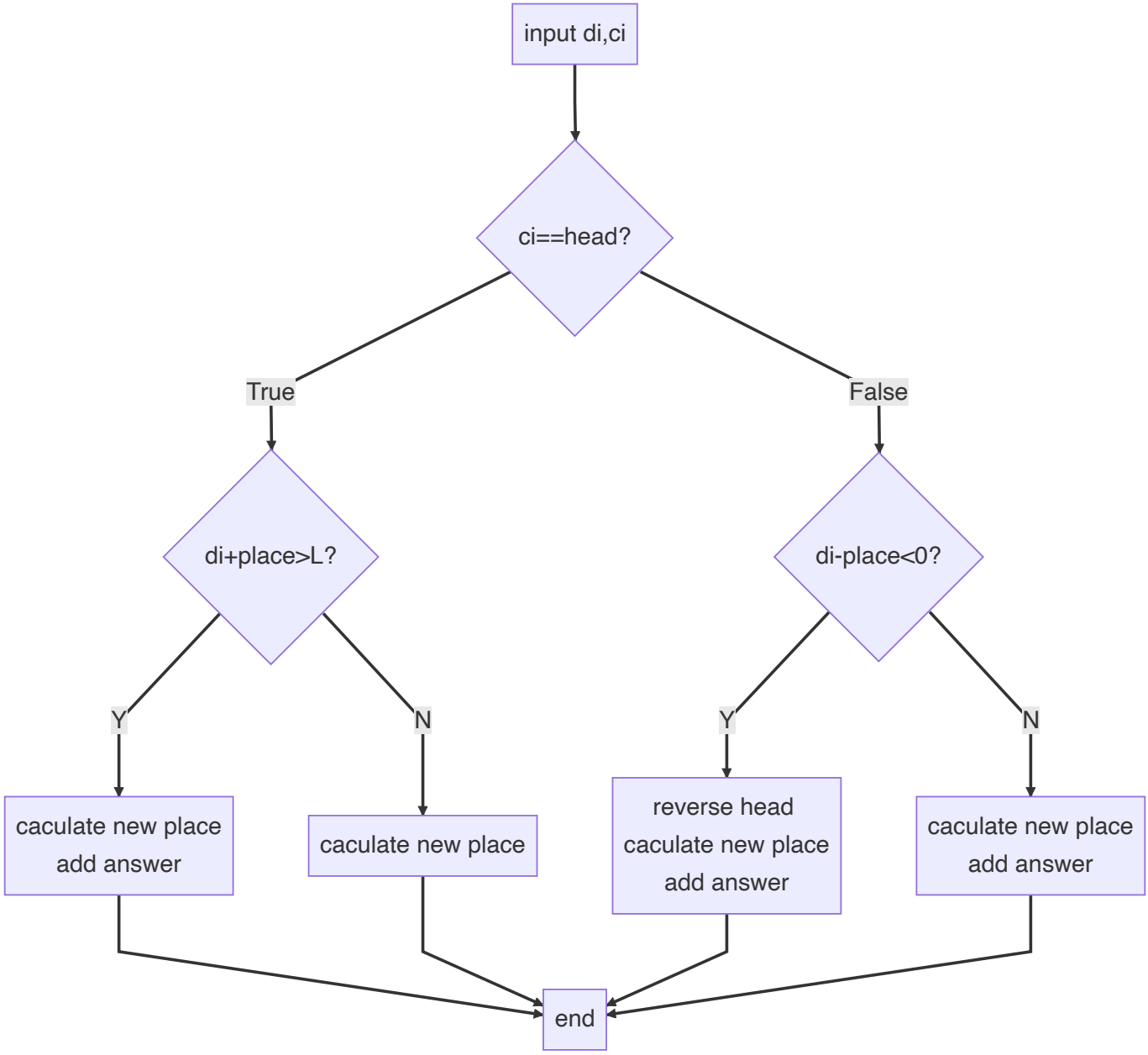
每组数据输出一个结果，每个结果占一行。

结果表示为 `Case #x: y`，其中 x 为组别编号（从 1 开始），y 为一个非负整数，表示机器最终记录的圈数。

时/空限制：	1s / 64MB
总通过数：	1279
总尝试数：	3851
来源：	Google Kickstart2022 Round H Problem A
算法标签	

该题的思路是`模拟+分类讨论`，每一组数据，设置一个`head`表示上次越过机器时的朝向，`place`表示在当前朝向下相对于起点的相对位置。“当前朝向下”这样处理可以使得place永远为正，且使顺时针和逆时针的处理相同，减少讨论次数。

根据本次方向和`head`是否一致进行分类讨论，每一次讨论继续分为“越界”和“非越界”进行讨论。



⚠️ 注意：数据量使得结果可能超出`INT_MAX`，故使用`long long int`存储结果。

Code:

```
//
//  main.cpp
//  4740-跑圈
//
//  Created by MacBook Pro on 2023/7/18.
//

#include <iostream>
using namespace std;

int T,L,N;
```

```
//是否是顺时针
bool check(char X){
    return X=='C';
}

int main() {
    scanf("%d",&T);
    for(int index=1;index<=T;index++){
        scanf("%d%d",&N,&L);
        int head=-1;    //面朝方向 1-顺时针 0-逆时针
        int place=0;    //位置
        long long int ans=0;
        for(int i=0;i<L;i++){
            int di;
            char ci;
            cin>>di>>ci;
            if(head==-1){
                head=check(ci);
                ans=(di/N);
                place=di%N;
            }
            else if(head==check(ci)){
                //方向相同
                if(place+di>=N){
                    //需要越过机器
                    ans+=(di-(N-place))/N+1;
                    place=(place+di)%N;
                }
                else{
                    //不越过机器
                    place=(place+di);
                }
            }
            else if(head!=check(ci)){
                //方向不同
                int circle=di-place;
                if(circle<0){
                    //虽然反向但还是在机器后
                    place-=di;
                }
                else{
                    //反向后越过机器
                    head=1-head;
                    ans+=(circle/N);
                    place=circle%N;
                }
            }
        }
```

```
        cout<<"Case #"<<index<<": "<<ans<<endl;
    }
}
```

代码提交状态：Accepted