Problem: *Counting Valleys*

Gary is an avid hiker. He tracks his hikes meticulously, paying close attention to small details like topography. During his last hike, he took exactly  steps. For every step he took, he noted if it was an *uphill* or a *downhill* step. Gary's hikes start and end at sea level. We define the following terms:

* A *mountain* is a non-empty sequence of consecutive steps *above* sea level, starting with a step *up* from sea level and ending with a step *down* to sea level.
* A *valley* is a non-empty sequence of consecutive steps *below* sea level, starting with a step *down* from sea level and ending with a step *up* to sea level.

Given Gary's sequence of *up* and *down* steps during his last hike, find and print the number of *valleys* he walked through.

**Input Format**

The first line contains an integer, , denoting the number of steps in Gary's hike.   
The second line contains a single string of  characters. Each character is  (where U indicates a step *up*and D indicates a step *down*), and the  character in the string describes Gary's  step during the hike.

**Constraints**

**Output Format**

Print a single integer denoting the number of valleys Gary walked through during his hike.

**Sample Input**

8

UDDDUDUU

**Sample Output**

1

**Explanation**

If we represent \_ as sea level, a step up as /, and a step down as \, Gary's hike can be drawn as:

\_/\ \_

\ /

\/\/

It's clear that there is only one valley there, so we print  on a new line.

*Solution:*

int main()

{

long steps;

long level,valleyTracker=0;

int checksum=0, initiate=0;

//Initiate tracks the direction of first step from the sea level //checksum calculated when sea level is reached based on up and down steps

cin>>steps;

char trek [steps]; //the trek array stores the direction of steps in Up and D characters

int convert[steps]={0};

//the convert array is the translated array of steps into 1 and -1 denoting the direction

/\*Feeding the data\*/

for(int i=0; i<=steps; i++)

{

if(i<steps)

{ cin>>trek[i]; //picks up the data from the console and converts at the same time

(trek[i]=='U' ? convert[i]=1 : convert[i]=-1);

}

/\*The parallel processing unit\*/

if(i>0)

{ if(i==1)

{ initiate=convert[0]; checksum=initiate; }

else //This loop starts form i=2 //forward motion applied

{

checksum+=convert[i-1];

if ( checksum==0 && initiate==-1)

//returns to ground level from the valley

{

valleyTracker+=1;

initiate=convert[i];

}

else if (checksum==0)

{ initiate=convert[i]; }

}

}

}

cout<<valleyTracker;

return 0;

}

**Elegant Solution: (more Efficient and cleaner)**

long steps;

long valleyTracker=0; //Counts the number of Valleys trekked

int tracker=0; //tracker counts the numbers of steps down and up in numbers to detect the sea lever reach point

cin>>steps;

char trek[steps]; //the trek array stores the direction of steps in Up and D characters

/\*Feeding the data\*/

For (int i=0; i<steps; i++)

{ cin>> trek[i];

(trek[i]=='U' ? tracker+=1 : tracker-=1);

(tracker==0 && trek[i]=='U' ? valleyTracker+=1 : valleyTracker+=0);

}

cout<<valleyTracker;

return 0;

}

* Anshul Aggarwal