**Ticket Counter :-**

Easy Accuracy: 56.39% Submissions: 11K+ Points: 2

**N** people from 1 to N are standing in the queue at a movie ticket counter. It is a weird counter, as it distributes tickets to the **first K** people and then the **last K** people and again first K people and so on, once a person gets a ticket moves **out** of the queue. The task is to find the **last** person to get the ticket.

**Example 1:**

**Input:**  
N = 9  
K = 3  
**Output:**  
6  
**Explanation:**  
Starting queue will like {1, 2, 3, 4, 5, 6, 7, 8, 9}. After the first distribution queue will look like {4, 5, 6, 7, 8, 9}. And after the second distribution queue will look like {4, 5, 6}. The last person to get the ticket will be 6.

**Example 2:**

**Input:**  
N = 5  
K = 1  
**Output:**  
3  
**Explanation:**  
Queue start as {1, 2, 3, 4, 5} -> {2, 3, 4, 5} -> {2, 3, 4} -> {3, 4} -> {3}  
Last person to get ticket will be 3.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function distributeTicket() which takes N and K as inputs and returns the last person to get the ticket.

**Expected Time Complexity:** O(N)  
**Expected Space Complexity:**O(N)  
Try to solve it using O(1) space complexity.

**Constraints:**  
1 <= K<= N <= 105

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

int distributeTicket(int n, int k) {

// code here

if(n==k) return n;

int start=0, end=n+1;

bool flag=false;

while(true){

//for first k people:-

for(int i=1; i<=k; i++){

++start;

if(start == end-1){

return start;

}

}

//for last k people

for(int i=1; i<=k; i++){

--end;

if(end == start+1){

return end;

}

}

}

}

};

//{ Driver Code Starts.

int main(){

int t;

scanf("%d ",&t);

while(t--){

int N;

scanf("%d",&N);

int K;

scanf("%d",&K);

Solution obj;

int res = obj.distributeTicket(N, K);

cout<<res<<endl;

}

}

// } Driver Code Ends

**T.C :- O(n)**

**S.C :- O(1)**