Batch: SY-IT(B3) **Experiment Number: 2**

Roll Number: 16010423076 **Name:Ritesh Jha**

Aim of the Experiment : To implement a Linear Search algorithm on hackerearth and optimize it’s solution using suitable data structures.

Program/ Steps:

Test cases :

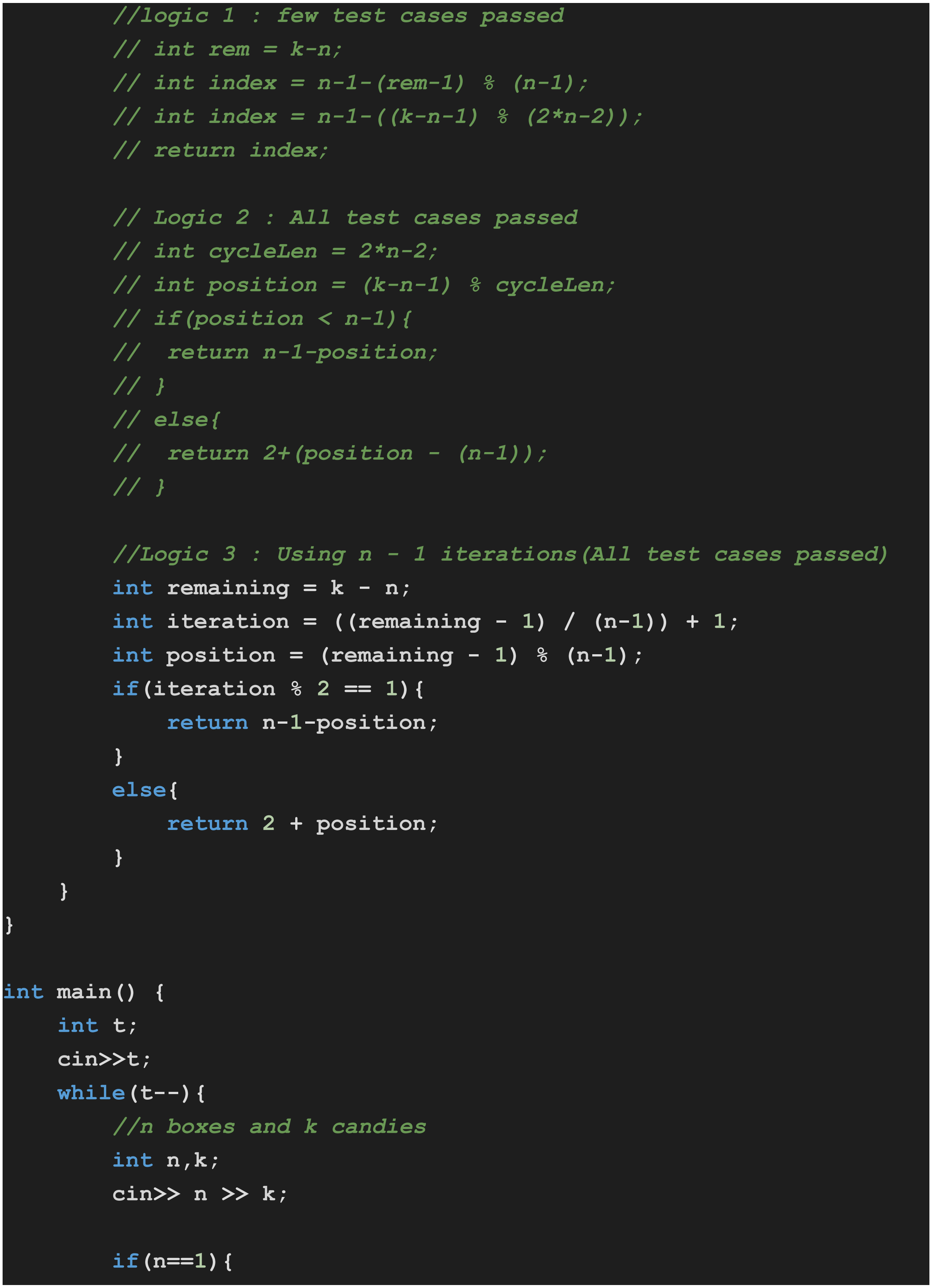
Sr. No.	**Sample Input**	**Sample Output**	**Description**	**Test Case Type**
1	5 7	3	Basic case with n < k and normal iterations	General
2	4 4	4	k is exactly equal to n	General
3	6 15	3		General

			k is more than n, verifying cycling pattern	
4	3 10	1	k is much larger than n, checking multiple cycles	General
5	1 1000000	1	Extreme case with n = 1, testing large k	Special
6	10 19	1	k just before completing a cycle	General
7	7 50	2	k significantly greater than n, testing multiple iterations	General

8	8 1000000000	7	Very large k value to test efficiency and cycle computation	Special
9	9 8	1	k is slightly larger than n, testing edge case handling	General
10	10 1	1	Minimum valid k, ensuring correct direct assignment	Special

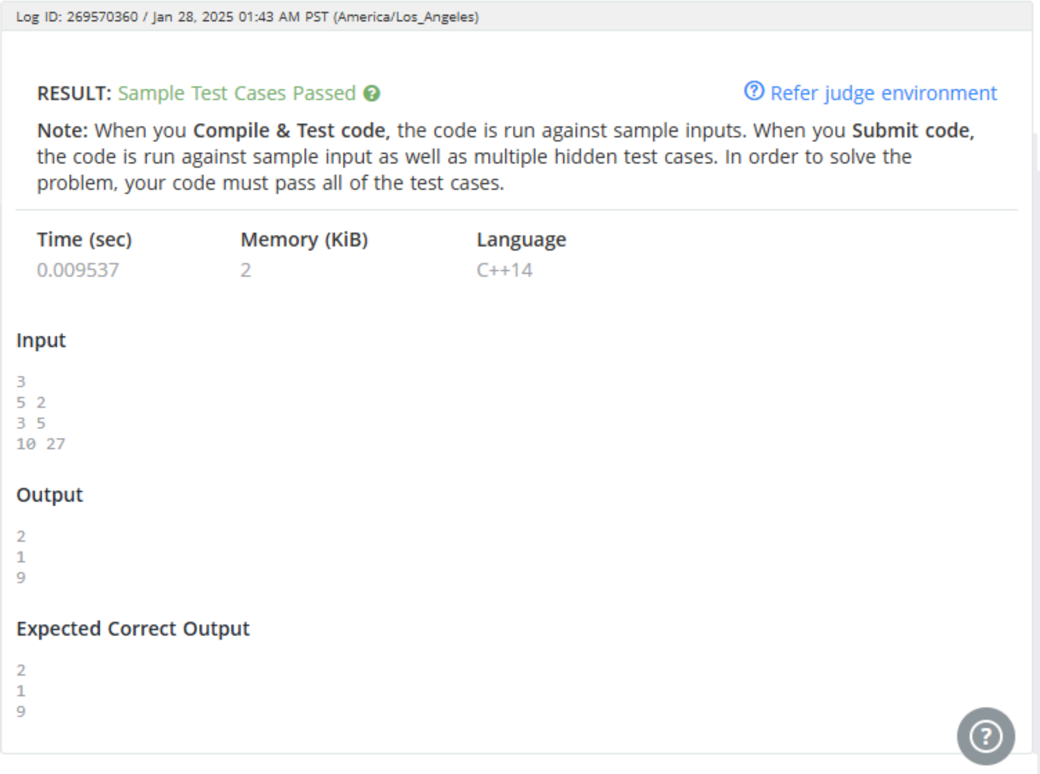
Code :

```
#include <iostream>
using namespace std;

int findboxindex(int n, int k){
    if(k <= n){
        return k;
    }
    else{
```



```
        //logic 1 : few test cases passed
        // int rem = k-n;
        // int index = n-1-(rem-1) % (n-1);
        // int index = n-1-((k-n-1) % (2*n-2));
        // return index;

        // Logic 2 : All test cases passed
        // int cycleLen = 2*n-2;
        // int position = (k-n-1) % cycleLen;
        // if(position < n-1){
        //  return n-1-position;
        // }
        // else{
        //  return 2+(position - (n-1));
        // }

        //Logic 3 : Using n - 1 iterations(All test cases passed)
        int remaining = k - n;
        int iteration = ((remaining - 1) / (n-1)) + 1;
        int position = (remaining - 1) % (n-1);
        if(iteration % 2 == 1){
            return n-1-position;
        }
        else{
            return 2 + position;
        }
    }
}

int main() {
    int t;
    cin>>t;
    while(t--){
        //n boxes and k candies
        int n,k;
        cin>> n >> k;

        if(n==1){
            cout<<1<<endl;

            //continue for other test cases
            continue;
        }
        int boxIndex = findboxindex(n,k);
        cout<<boxIndex<<endl;
    }
}
```

Output/Result:



Log ID: 269570360 / Jan 28, 2025 01:43 AM PST (America/Los_Angeles)

RESULT: Sample Test Cases Passed

Note: When you **Compile & Test code**, the code is run against sample inputs. When you **Submit code**, the code is run against sample input as well as multiple hidden test cases. In order to solve the problem, your code must pass all of the test cases.

Time (sec)	Memory (KiB)	Language
0.009537	2	C++14

Input

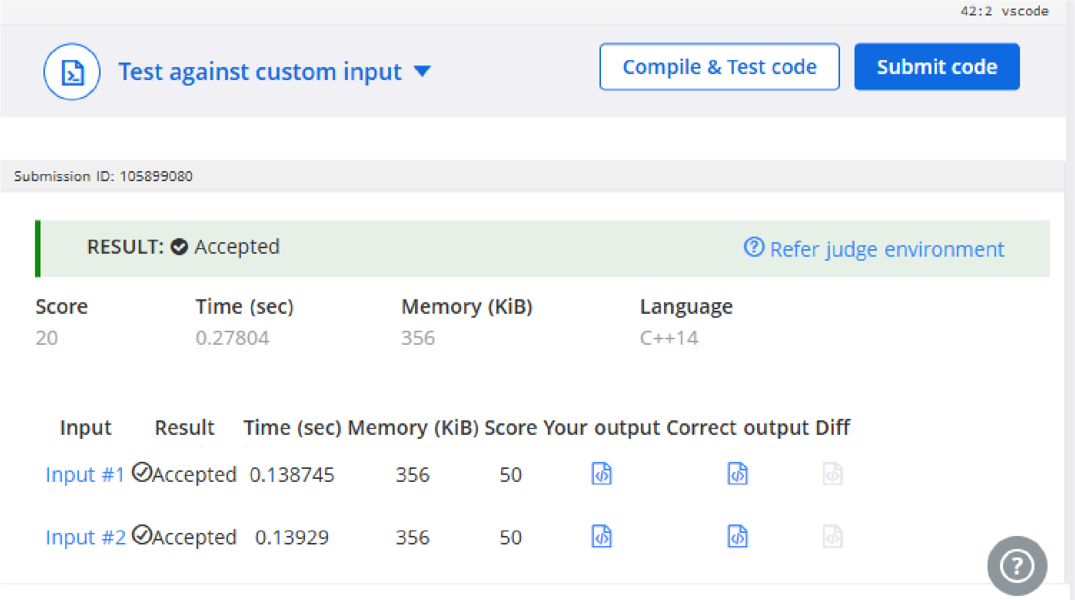
```
3
5 2
3 5
10 27
```

Output

```
2
1
9
```

Expected Correct Output

```
2
1
9
```



Submission ID: 105899080

RESULT: Accepted

Score	Time (sec)	Memory (KiB)	Language
20	0.27804	356	C++14

Input	Result	Time (sec)	Memory (KiB)	Score	Your output	Correct output	Diff
Input #1	Accepted	0.138745	356	50			
Input #2	Accepted	0.13929	356	50			

Outcomes: CO1. Inculcate the best practices that are essential for competitive programming

Conclusion (based on the Results and outcomes achieved):

From this experiment, I learned how to systematically determine the placement of candies in boxes using a repeating pattern of forward and backward passes. I tried to implement the code using 3 different logics to calculate the index when the number of candies was greater than the number of boxes. By breaking down the problem into smaller steps such as identifying the iteration and calculating the position within that iteration I was able to create a clear and efficient solution.

References:

1. https://www.hackerearth.com/practice/data-structures/arrays/1-d/practice-problems/algorithm/sum-as-per-frequency-88b00c1f/
2. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
3. Antti Laaksonen, “Guide to Competitive Programming”,Springer,2018
4. Gayle Laakmann McDowell,” Cracking the Coding Interview”,CareerCup LLC,2015
5. Steven S. Skiena Miguel A. Revilla,”Programming challenges, The Programming Contest Training Manual”, Springer, 2006
6. Antti Laaksonen, “Competitive Programmer’s Handbook”, Hand book, 2018

7. Steven Halim and Felix Halim, “Competitive Programming 3: The Lower Bounds of Programming Contests”, Handbook for ACM ICPC