## **MNC COMPANIES - SET 001**

Solved Challenges 3/10



Back To Challenges List



# function countHardPrograms

#### ID:12075 Solved By 44 Users

#### Accenture

You are required to complete the given code. You can click on Run anytime to check the compilation/execution status of the program. You can use printf to debug your code. The submitted code should be logically/syntactically correct and pass all test cases. Do not write the main() function as it is not required.

Code Approach: For this question, you will need to complete the code as in the given implementation. We do not expect you to modify the approach.

The function/method **countHardPrograms** accepts three arguments - **N**, **arr** and **K**, an integer representing the number of chapters in a competitive programming book, an array of integers representing the number of programs in the N chapters and an integer K representing the maximum number of programs a page can hold.

The format of the competitive programming book is as follows.

- The N chapters are numbered from 1 to N (The chapter 1 always starts from the page 1).
- The programs in each chapter are numbered from 1 to X, where X is the number of programs in a chapter.
- Each page can hold up to K programs. Only the last page of a chapter may contain less than K programs.
- Each chapter starts on a new page, so a page will never contains programs from more than one chapter.

The function/method countHardPrograms must return an integer representing the number of hard programs in the book. The hard programs are identified based on the following condition.

- If there is a program on a page whose page number is equal to its number(the program's number), then the program is called a hard program.

Your task is to complete the code in the function **countHardPrograms** so that it passes all the test cases.

## **Boundary Condition(s):**

1 <= N, K, Each integer value <= 100

# **Example Input/Output 1:**

Input:

5 3

421103

Output:

3

## Explanation:

The 1<sup>st</sup> chapter contains 4 programs.

The **2<sup>nd</sup>** chapter contains **2** programs.

The **3<sup>rd</sup>** chapter contains **1** program.

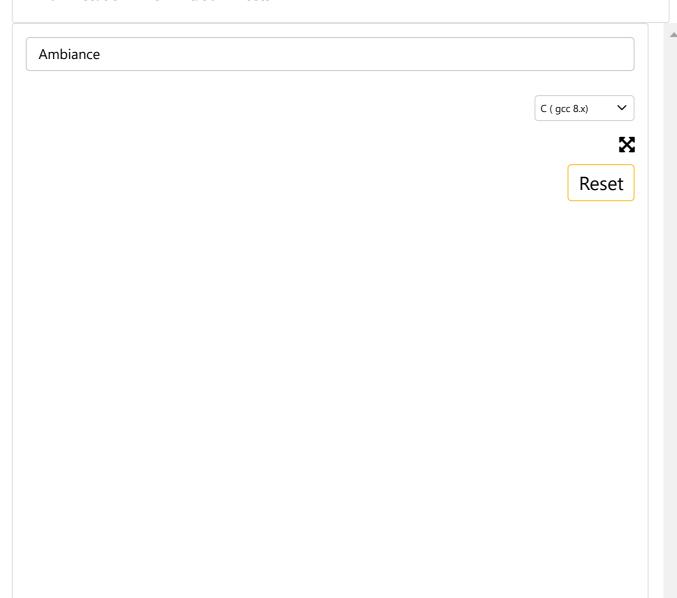
The **4**<sup>th</sup> chapter contains **10** programs.

The **5**<sup>th</sup> chapter contains **3** programs.

Each page can hold K = 3 programs.

The 1<sup>st</sup> page contains the programs 1, 2, 3 (chapter 1). The **2<sup>nd</sup>** page contains the program **4** (chapter 1). The **3<sup>rd</sup>** page contains the programs **1**, **2** (chapter 2). The **4**<sup>th</sup> page contains the program **1** (chapter 3). The **5<sup>th</sup>** page contains the programs **1**, **2**, **3** (chapter 4). The **6<sup>th</sup>** page contains the programs **4**, **5**, **6** (chapter 4). The **7<sup>th</sup>** page contains the programs **7**, **8**, **9** (chapter 4). The **8**<sup>th</sup> page contains the program **10** (chapter 4). The **9<sup>th</sup>** page contains the programs **1**, **2**, **3** (chapter 5). The **3** hard programs are given below. Chapter 1: 1st program (Page number is 1). Chapter 4: 6<sup>th</sup> program (Page number is 6). Chapter 4: 7<sup>th</sup> program (Page number is 7). **Example Input/Output 2:** Input: 6 5 6 3 4 13 5 14 Output: 7

### **Max Execution Time Limit: 50 millisecs**



```
int countHardPrograms(int N, int *arr, int K)
    int page_no=1;
    int count=0;
    for(int index=0;index<N;index++)</pre>
        int count_prob=1;
        // printf("%d \n",arr[index]);
        while(count prob<=arr[index])</pre>
        {
             for(int k=1;k<=K && count_prob<=arr[index];k++)</pre>
                 if(page no==count prob)
                     count++;
                 // printf("%d %d %d \n",count prob,page no,count);
                 count_prob++;
             page_no+=1;
        }
    }
   return count;
}
```

```
421
  2
  131
  2 3 1
  141
  10
  151
  251
  3 5 1
  461
  561
  662
  773
  873
  973
  1083
  3
  193
  293
  393
 Save
          Run
☐ Run with a custom test case (Input/Output)
```