Algorithm:

We must solve this problem using divide and conquer approach so we create a function that recursively calls itself and another function and in each call, the function halves the array of tasks until reaches to the single indexes of the array. The main concept of the algorithm is to divide the array into two halves and check the possibility of each half. But the main problem for solving this problem is how to check the possibility of the middle arrays (the arrays that starts from before of index m and ends after the index number m) which will create in every round that the functions are being called. So in case we solve this problem, actually we will solve the main problem entirely.

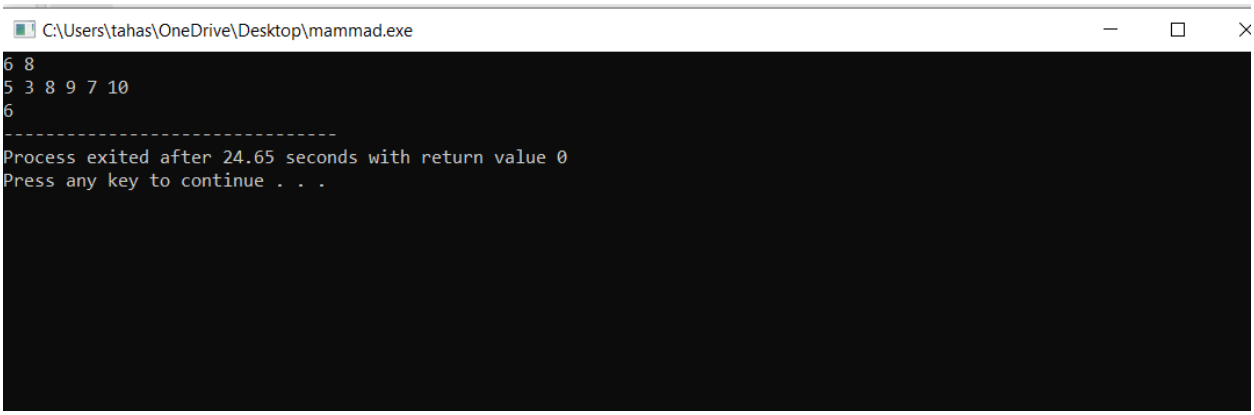
So we create below function which recursively calls itself twice a round to calculate the possibilities of the left half and the right half and as well it calls another function name “crossingarray” which calculates the arrays that crosses the middle index (m).

```
35  }
36
37  int findarray(int arr[], int l, int h, int k)
38  {
39      if (l==h){
40          return 0;
41      }
42
43      int m = (l + h) / 2;
44
45      return findarray(arr, l, m ,k)+findarray(arr, m + 1, h, k)+Crossingarray(arr, l, m, h, k);
46  }
47
```

So in case of checking this possibility of the crossing arrays, we define a block of code in line number 13 to 30. In this block actually we are merging two arrays. One which starts from index l to m and another one which starts from index m+1 to h. In nested loop first we start from the last index of left array and in the inner loop we define the range of right array and in line number 19 to 23 we find the bigest number in the specific range. We want mammad to do this task by himself, so in case of removing this task from other tasks we have to find it every time that our range changes. If the sum of the conent of array except of the bigest task in the specifide range is the mod of the number of the developers, we take this as a right possibility and increase the counter (z) which we want to return as the output of the function. Finally the sum of these z’s is the answer of this problem.

```
int Crossingarray(int arr[], int l, int m, int h,int k)
{

    int j,i,p,q,sum = 0,z=0;
    int maxnum = arr[l];



    for(i=m;i>=l;i--){
        for(p=i;p<=m;p++){
            sum=sum+arr[p];
        }
        maxnum=arr[i];
        for(j=m+1;j<=h;j++){
            for(q=i+1;q<=j;q++){
                if(maxnum<arr[q]){
                    maxnum=arr[q];
                }
            }
            sum=sum+arr[j];
            if((sum-maxnum)%k==0){
                z++;
            }
        }
        sum=0;
    }

    return z;


}
```

Example:



In this example the possible arrays are:

```
5 3 8
9 7 10
8 9
5 3 8 9
5 3 8 9 7 10
8 9 7 10
```