LAB -7

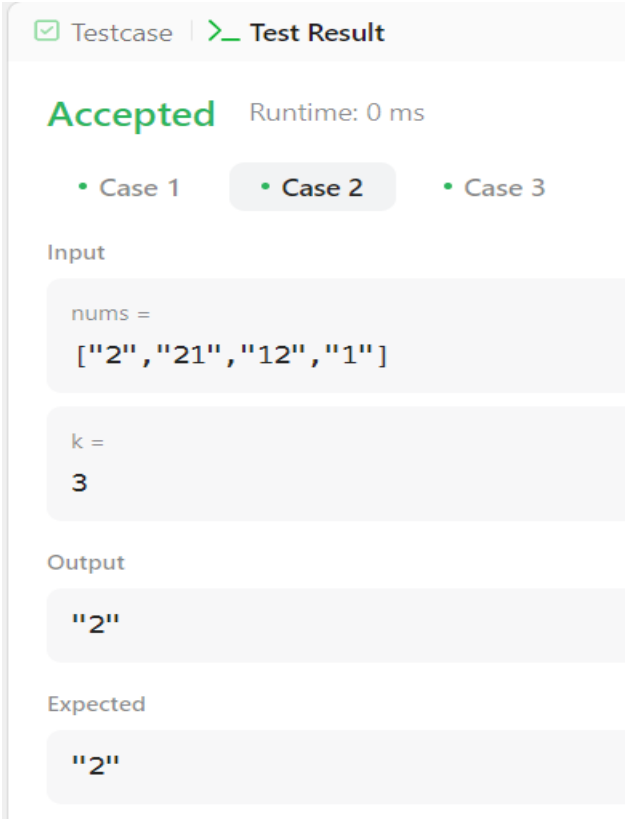
Question : 1. Leetcode exercises on find the Kth largest Integer in the array

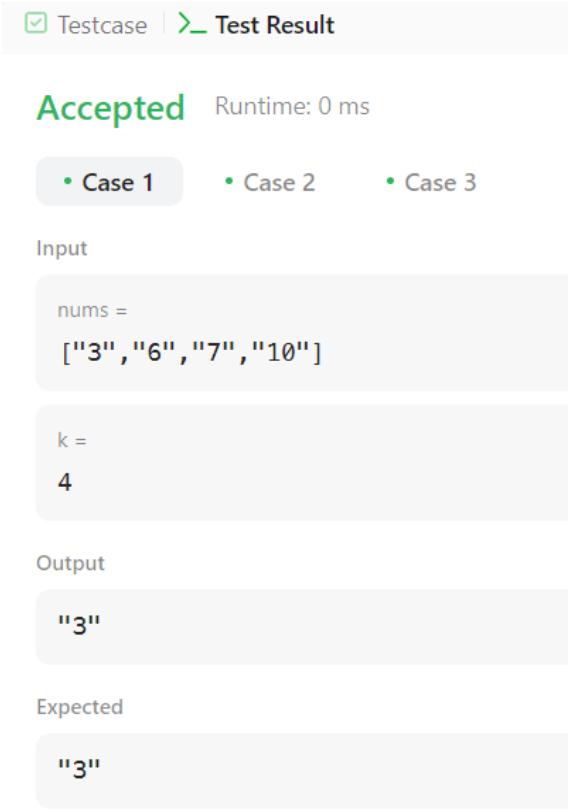
2. Program on Johnson Trotter Algorithm

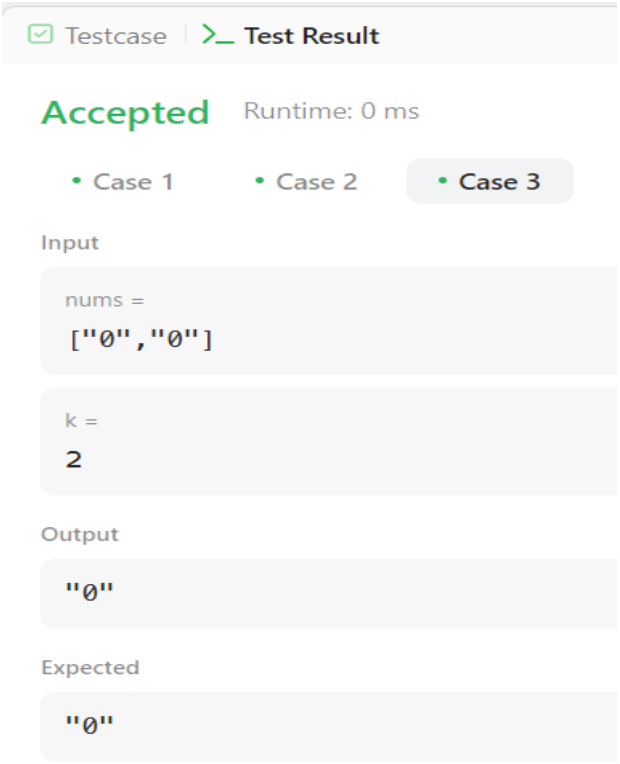
1. SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void bsort(char **nums, int n){
   for(int i = 0; i < n-1; i++){
      for(int j = i+1; j<n; j++){
         int x = atoi(nums[i]);
         int y = atoi(nums[j]);
         if(x > y){
            char *temp = nums[i];
            nums[i] = nums[j];
            nums[j] = temp;
         }
      }
   }
}

char* kthLargestNumber(char** nums, int numsSize, int k) {
   bsort(nums, numsSize);
   return nums[numsSize - k];
}
```

RESULT:


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
nums =
["2","21","12","1"]
k =
3
Output
"2"
Expected
"2"





2. SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
```

```
// Global variable to count flags (not necessary, but keeping it as per
original)
int flag = 0;

// Function to swap two integers
void swap(int *a, int *b)
{
   int t = *a;
   *a = *b;
   *b = t;
}

// Function to find the mobile integer in the array
int find_Mobile(int arr[], int d[], int num)
{
   int mobile = 0;
   int mobile_p = 0;
   int i;

   for (i = 0; i < num; i++)
   {
      if ((d[arr[i] - 1] == 0) && (i != 0))
      {
         if (arr[i] > arr[i - 1] && arr[i] > mobile_p)
         {
            mobile = arr[i];
            mobile_p = mobile;
         }
         else
         {
            flag++;
         }
      }
      else if ((d[arr[i] - 1] == 1) && (i != num - 1))
      {
         if (arr[i] > arr[i + 1] && arr[i] > mobile_p)
         {
            mobile = arr[i];
```

```
            mobile_p = mobile;
          }
          else
          {
            flag++;
          }
        }
        else
        {
          flag++;
        }
    }

    if ((mobile_p == 0) && (mobile == 0))
      return 0;
    else
      return mobile;
}

// Function to find the position of a given integer in the array
int search(int arr[], int num, int mobile)
{
    int g;
    for (g = 0; g < num; g++)
    {
      if (arr[g] == mobile)
        return g + 1;
      else
      {
        flag++;
      }
    }
    return -1;
}

// Function to generate permutations using Johnson Trotter algorithm
void permutations(int arr[], int d[], int num)
{
```

```
    int mobile = find_Mobile(arr, d, num);
    int pos = search(arr, num, mobile);

    if (d[arr[pos - 1] - 1] == 0)
        swap(&arr[pos - 1], &arr[pos - 2]);
    else
        swap(&arr[pos - 1], &arr[pos]);

    for (int i = 0; i < num; i++)
    {
        if (arr[i] > mobile)
        {
            if (d[arr[i] - 1] == 0)
                d[arr[i] - 1] = 1;
            else
                d[arr[i] - 1] = 0;
        }
    }

    for (int i = 0; i < num; i++)
    {
        printf(" %d ", arr[i]);
    }
}

// Function to calculate factorial of a number
int factorial(int k)
{
    int f = 1;
    for (int i = 1; i <= k; i++)
    {
        f = f * i;
    }
    return f;
}

int main()
{
```

```
   int num = 0;

   printf("Johnson trotter algorithm to find all permutations of given
numbers \n");
   printf("Enter the number\n");
   scanf("%d", &num);

   int arr[num], d[num];
   int z = factorial(num);

   printf("Total permutations = %d\n", z);
   printf("All possible permutations are: \n");

   for (int i = 0; i < num; i++)
   {
      d[i] = 0;
      arr[i] = i + 1;
      printf(" %d ", arr[i]);
   }

   printf("\n");

   for (int j = 1; j < z; j++)
   {
      permutations(arr, d, num);
      printf("\n");
   }

   return 0;
}
```

RESULT:

```
Johnson trotter algorithm to find all permutations of given numbers
Enter the number
123
Total permutations = 0
All possible permutations are:
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92
93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  11
8  119  120  121  122  123

Process returned 0 (0x0)   execution time : 1.918 s
Press any key to continue.
```