1) Johnson trotter. 13/6/24

```
#include<stdio.h>
#include<stdlib.h>
int flag=0; int swap(int *a, int *b)
{
  int t= *a;
  *a = *b;
  *b = t;
}
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;
}
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];
          } else {
             mobile_p = mobile;
             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;
            flag++
          } else {
             flag++;
```

```
    } else
    {
      flag++;
    }
  }
  if ((mobile_p == 0) && (mobile == 0))
    return 0;
  else
    return mobile;
}
void permutations (int arr[], int d[], int num)
{
    int i;
    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(i=0; i<num; i++)
        {
            printf("%d", arr[i]);
        }
}
int factorial(int k)
{
    int f=1;
    int i=0;
    for(i=1; i<k+1; i++)
    {
        f = f*i;
    }
    return f;
}
```

```
int main ( )
{
  int num= 0;
  int i;
  int j;
  int z =0;
  printf (" Johnson Trotter algorithm to find all permutati
     - ons of given numbers \n");
  printf(" Enter the number \n");
  scanf (" %d" , &num);
  int arr[num], d[num];
  z = factorial (num);
  printf(" total permutations = %d", z);
  printf(" \n All possible permutations are : \n");
  for (i=0; i<num; i++)
  {
     d[i] =0;
     arr[i] = i+1;
     printf(" %d ", arr[i]);
  }
  printf ("\n");
  for (j=1; j<z; j++)
  {
       permutations(arr, d, num);
       printf("\n");
  }
  return 0;
}
```

o/p :-

Johnson Trotter algorithm

Enter the number

4

total permutations = 24

All possible permutations are :-

```
1 2 3 4
1 2 4 3
1 4 2 3
4 1 2 3
4 1 3 2
1 4 3 2
1 3 4 2
1 3 2 4
3 1 2 4
3 1 4 2
```

```
4 3 1 2
4 3 2 1
3 4 2 1
3 2 4 1
3 2 1 4
2 3 1 4
2 3 4 1
2 4 3 1
4 2 3 1
4 2 1 3
2 4 1 3
```

2134

Pattern Matching

2)

```
#include <stdio.h>
#include <string.h>
int substringMatch(char *text, char *pattern){
    int textLength = strlen(text);
    int patternLength = strlen(pattern);

    for(int i=0; j<= textLength - patternLength; i++)
    {
        int j;
        for(j=0; j<patternLength; j++)
        {
            if(text[i+j] != pattern[j])
                break;
        }
        if(j == patternLength)
            return i;
    }
    return -1;
}
int main(){
    char text[100], pattern[100];
    printf("Enter the main text: ");
    scanf("%s", text);
    printf("Enter the pattern to search: ");
    scanf("%s", pattern);
    int index = substringMatch(text, pattern);
    if(index != -1)
        printf("Substring found at index: %d\n", index);
    else
        printf("Substring not found.\n");
    return 0;
}
```

o/p:

Enter the main text: Run_world
Enter the pattern to search: world
Substring found at index: 4

3) Find the Kth Largest integers in the Array.

```
int cmp(const void *a, const void *b){
    const char *str1 = *(const char **)a;
    const char *str2 = *(const char **)b;
    if (strlen(str1) == strlen(str2)){
        return strcmp(str1, str2);
    }
    return strlen(str1) - strlen(str2);
}
char *KthLargestNumber(char **nums, int numsSize, int k){
    qsort(nums, numsSize, sizeof(char *), cmp);
    return nums[numsSize - k];
}
```

o/p:

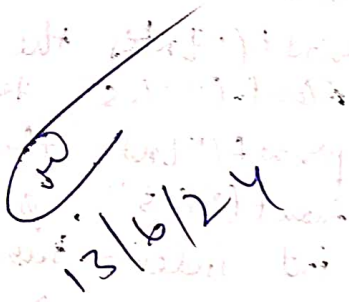
input

nums = ["3", "6", "7", "10"]

case 1: K = 4

output = "3"

Expected = "3".



case 2: nums = ["2", "21", "12", "1"]

K = 3

o/p:

2

Expected

2

case 3: nums = ["0", "0"]

K = 2.

o/p: 0

Expected: 0