⇒ Johnson Trotter : - LAB (7) 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(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;
}
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(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, j, z=0;
  printf("To find all permutations of given nos \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 perm. are: \n");
  for(i=0; i<num; i++)
  { d[i]=0;
    arr[i]=i+1;
    printf("%d", arr[i]);
  }
  for(j=1; j<z; j++)
  { permutations(arr, d, num);
    printf("\n");
  }
  return 0;
}
```

output:

```
Enter the number: 3
total permutations = 6
All possible permutations are:
1 2 3
1 3 2
3 1 2
3 2 1
2 3 1
2 1 3
```

Enter the number: 4
total permutations = 24
All possible permutations are:

1 2 3 4	1342	4321	2431
1 2 4 3	1324	3421	4231
1 4 2 3	3124	3241	4213
4 1 2 3	3142	3214	2413
4 1 3 2	3412	2314	2143
1 4 3 2	4312	2341	2134

Pattern Matching:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int patternMatch(int n, int m, char text[], char pattern[])
{
  int i,j;
  for(i=0; i<=(n-m); i++)
  {
    j=0;
    while(j<m && text[i+j]==pattern[j])
    {
      j++;
    }
    if(j==m)
      return i;
  }
  return -1;
}
int main()
{
  int m,n;
  char text[100];
  char pattern[50];
  printf("Enter length of text:\n");
  scanf("%d",&n);
  printf("Enter length of pattern:\n");
  scanf("%d",&m);
```

```
printf("Enter the text:\n");
scanf("%d", &n);
printf("Enter the pattern:\n");
scanf("%s", pattern);
printf("Enter the text:\n");
scanf("%s", text);
int k = patternMatch(n, m, text, pattern);
if (k != -1) {
    printf("pattern matches from position %d", k+1);
}
else {
    printf("pattern doesn't match with text given");
}
return 0;
}
```

(J)

Output:

Enter length of text : 10
Enter length of pattern : 5
Enter the text : helloworld
Enter the pattern : world
pattern matches from position 6.

Leetcode - (3) - Find kth largest Integer 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];
}
```

output:

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

k = 4

=> "3"

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

k = 3

=> "2"

nums = ["0", "0"]

k = 2

=> "0"

13/6/24