13/06/21

Johnson Trotter program.

```
#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=0;
for( i=0 ; i<num ; i++)
  {
    if ((d[ass(i)-1) ==0) && i!=0)
    {
      if (ass(i)> ass[i-1] && ass(i)>
      mobile_p ) {
        mobile =ass[i];
        mobile_p = mobile;

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

void permutation (int ass(), int d[], int num)
  {
    int i;
    int mobile = find_Mobile (ass, d, num);
    int pos = search (ass, num, mobile);
    if (d[ass[pos-1]] == ass[pos-2]);
    else
      swap (&ass[pos-1], &ass[pos]);
    for(int i=0 ; i<num ; i++) {
      if( ass[i] >mobile ) {
        if( d[ass(i)] == -1 )
        else
          d[ass[i]-1] = 0;

int main ()
{
```

```
int num=0, i, j, z=0;
printf("Johnson trotter algo to find all
        permutations of no.");
printf("Enter the no");
Scanf("%d", &num);
z = factorial(num);
for(i=0; i<=num; i++) {
    d[i]=0;
    arr[i] = i+1;
    printf("%d", arr[i])
}
return 0;
}
```

Output :-

Enter the no : 3
All possible permutations are:
1 2 3
1 3 2
3 1 2
3 2 1
2 3 1
2 1 3

Enter the no: 4
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 2 1 4

3 4 2 1

3 4 1 2

4 3 1 2

4 3 2 1

2 3 1 4

2 4 3 1

2 4 1 3

2 1 3 4

Substring Matching program

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

void patternmatch(char *text, char *pattern)
{
    int textmatch = strlen(text);
    int patternlength = strlen(pattern);

    for(int i=0; i<= textlength - patternlength;
    {
        int j;
        for(j=0; j < patternlength; j++) {
            if (text[i+j] != pattern[j])
                break;
        }
        if (j == patternlength)
            fprintf("Pattern found at index : %d \n",
    }
}
}

int main()
{
    char text[100];
    char pattern[100];

    printf("Enter the text :");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text, "\n")] = '\0';
```

pattern_match(text, pattern);

return 0;
}

Output:

+ Enter the text: hi my name is rohit

Ent the pattern: my

i++)

Pattern found at index: 3

+ Ent the text: Hellow world

Ent the pattern: world

Pattern found at index: 7

+ leetcode

+ Find the K^th 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:-

Case 1:

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

K = 4

Case 2

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

K = 3

Case 3

nums = ["0", "0"]

K = 2

13/6/2024