1) JOHNSON TROTTER ALGORYTHM

```
#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_f)
        {
                mobile = arr[i];
        mobile_f = mobile;
        }
        else
        {
                flag ++;
        }
    }
    else if ( d[arr[i]-1] == 1) && i != num-1)
    {
         if (arr[i] > arr[i+1] && arr[i] > mobile_f)
         {
                mobile = arr[i];
                mobile_f = mobile;
         }
         else
         {
                flag ++;
         }
    }
    else
    {
           flag ++;
     }
   }
   if((mobile_f == 0) && (mobile == 0))
                  return 0;
   else
                  return mobile;
   }
```

[illegible] 13/6/24

```
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-1]);
    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; i++)
    {
        f= f*i;
    }
```

```
        return f;
    }
int main()
{
        int num = 0;
        int i;
        int j;
        int z = 0;
        printf("Johnson trotter algorythm to find
                all permutations of given number\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);
                print f("\n");
        }
        return 0;
    }
```

→ output

Johnson trotter algorythm to find all permutations of given numbers

```
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
```

② PATTERN MATCHING

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

int substring Match (const char *text, const char *
                                         pattern)
  int textlen = strlen (tex);
 int patternlen = strlen (pattern);

for (int i=0; i <= textlen - patternlen; i++)
{
        int j;
```

Date
Page

```
    for (j= 0; j <patternLen; j++)
    {
            if (text [i+j] != pattern [j])
                break;
    }
        if (j == patternlen)
                return i;
  }

  return -1;

int main()
{
      char text [100], pattern [100];
      printf ( " Enter the text :\n ");
      scanf ( " %s", text);
      printf ( " Enter the pattern to match: ");
      scanf ( " %s", pattern);

      int position = substringMatch ( text, pattern);

      if ( position != -1)
            printf ("Pattern found at position:
                  %d \n", position);
    else
            printf ( "Pattern not found in the
                  text. \n");
    return 0;
}
```

[illegible] 13/6/24

→ output

```
Enter the text: Hello_ world
Enter the pattern to match: World

Pattern fond at position 6
```