Q) Implement Johnson Trotter algorithm to generate permutations.

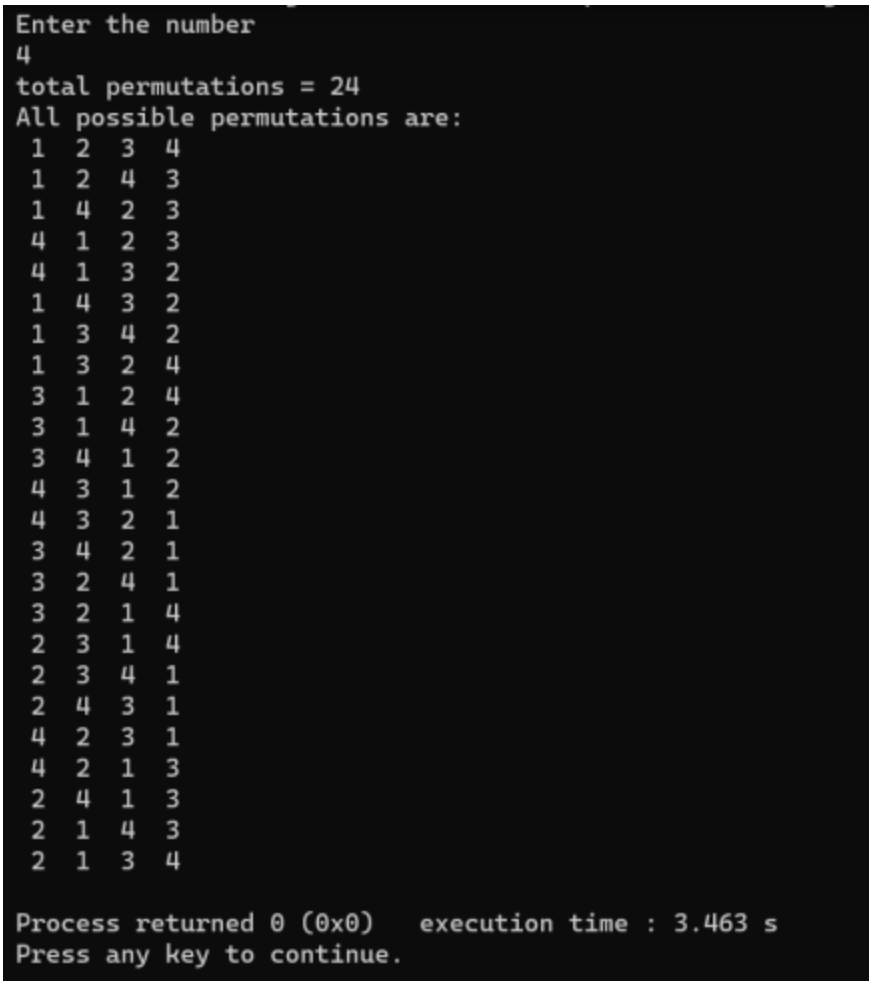
CODE-

```
#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_Moblie(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;
}
void permutations(int arr[],int d[],int num)
{
```

```
int i;
int mobile = find_Moblie(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 permutations 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("\nAll 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;
}
```



Q) write a c code for merge sort

```
#include<stdio.h>
void main(){
   int low=0,high,i,n;
   int arr[15];
   printf("Enter the number of elements in the array\n");
   scanf("%d",&n);
   high=n-1;
   printf("Enter the elements of the array\n");
   for(i=0;i<n;i++){
     scanf("%d",&arr[i]);
```

```
    }
    mergeSort(low,high,n,arr);
    printf("Sorted array is : ");
    for(i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
}

void mergeSort(int low,int high,int n,int arr[n])
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        mergeSort(low,mid,n,arr);
        mergeSort(mid+1,high,n,arr);
        merge(low,mid,high,n,arr);
    }
}

void merge(int low,int mid,int high,int n,int arr[n])
{
    int i=low,j=mid+1,k=low,c[n];
    while(i<=mid&&j<=high)
    {
        if(arr[i]<arr[j])
        {
            c[k]=arr[i];
            i++;
```

```
        k++;
      }
      else
      {
        c[k]=arr[j];
        j++;
        k++;
      }
    }
    while(i<=mid)
    {
      c[k]=arr[i];
      i++;
      k++;
    }
    while(j<=high)
    {
      c[k]=arr[j];
      j++;
      k++;
    }
    for (i = low; i <= high; i++)
    {
      arr[i]=c[i];
}
```

```
Enter the number of elements in the array
8
Enter the elements of the array
88 41 12 10 59 87 60 45
Sorted array is : 10 12 41 45 59 60 87 88
Process returned 8 (0x8)   execution time : 30.043 s
Press any key to continue.
```

Classmate
Date 13 July
Page 13

Implement Johnson Trotter algorithm to generate permutations.

```
#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 = mobile;
}
else
flag++;
}
else if ((d[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(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 x=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];
x=factorial(num);
printf("total permutations=%d", x);
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<x; j++){
permutations(arr, a, num);
printf("\n");
}
return 0;
}
```

Output

Johnson trotter algorithm to find all permutations of given numbers.

Enter the number.

4

Total permutations = 24

All possible permutations are

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

→ Write a c program to implement merge sort.

```
#include <stdio.h>
#include <stdlib.h>
void merge (int low, int mid, int high, int array [20], int
mer [20])
{
int i = low;
int j = mid + 1;
int k = 0;
while (i <= mid && j <= high)
{
 if (array [i] < array [j])
   {
   mer [k] = array [i];
        i++;
        k++;
   }
  else
   {
   mer [k] = array [j];
        j++;
        k++;
   }
 }
 while (i <= mid)
 {
  mer [k] = array [i];
    i++;
    k++;
 }
while (j <= high)
   {
   mer [k] = array [j];
      j++;
      k++;
```

```
}
while (j<=high)
{
mer[k] = array[j];
j++;
k++;
}
}
while (i<=mid)
{
mer[k] = array[i];
i++;
k++;
}
while (j<=high)
{
mer[k] = array[j];
j++;
k++;
}
for (int i=0; i<k; i++)
{
array[low+i] = mer[i];
}
}
void merge_sort (int low, int high, int array[20],
int merged[20])
{
if (low<high)
{
int mid = (low+high)/2;
merge_sort (low, mid, array, merged);
merge_sort (mid+1, high, array, merged);
merge (low, mid, high, array, merged);
```

```
}
}
int main()
{
int n, array[30];
Printf("Enter size:");
Scanf("%d", &n);
for(int i=0; i<n; i++)
{
Printf("Enter element:");
Scanf("%d", &array[i]);
}
int merged[30];
merge_sort(0, n-1, array, merged);
for(int i=0; i<n; i++)
{
Printf("%d", array[i]);
}
}
```

Output

```
Enter size: 5
Enter element: 2
Enter element: 8
Enter element: 1
Enter element: 10
Enter element: 18
1 2 8 10 18
```

12/7/23