3. Implement Johnson Trotter algorithm to generate permutations.

## Code:

```
#include<stdio.h>
#include<stdbool.h>
#define left_to_right true
#define right_to_left false
int getPosOfMobile(int a[], int n, int mobile) {
  for (int i = 0; i < n; i++) {
     if (a[i] == mobile)
        return i + 1;
  }
  return 0;
}
int getMobile(int a[], bool dir[], int n) {
  int mobile prev = 0, mobile = 0;
  for (int i = 0; i < n; i++) {
     // direction 0 represents RIGHT TO LEFT.
     if (dir[a[i] - 1] == right_to_left && i != 0) {
        if (a[i] > a[i - 1] && a[i] > mobile_prev) {
           mobile = a[i];
           mobile prev = mobile;
        }
     }
     // direction 1 represents LEFT TO RIGHT.
     if (dir[a[i] - 1] == left_to_right && i != n - 1) {
        if (a[i] > a[i + 1] && a[i] > mobile_prev) {
           mobile = a[i];
           mobile_prev = mobile;
        }
     }
  }
  if (mobile == 0 && mobile_prev == 0)
     return 0;
  else
     return mobile;
}
```

```
void produceOnePermutation(int a[], bool dir[], int n) {
  int mobile = getMobile(a, dir, n);
  int pos = getPosOfMobile(a, n, mobile);
  if (dir[a[pos - 1] - 1] == right_to_left) {
     int temp = a[pos - 1];
     a[pos - 1] = a[pos - 2];
     a[pos - 2] = temp;
  } else if (dir[a[pos - 1] - 1] == left_to_right) {
     int temp = a[pos];
     a[pos] = a[pos - 1];
     a[pos - 1] = temp;
  }
  // changing the directions for elements
  // greater than largest mobile integer.
  for (int i = 0; i < n; i++) {
     if (a[i] > mobile) {
        if (dir[a[i] - 1] == left_to_right)
           dir[a[i] - 1] = right_to_left;
        else if (dir[a[i] - 1] == right_to_left)
           dir[a[i] - 1] = left_to_right;
     }
  }
  for (int i = 0; i < n; i++)
     printf("%d ", a[i]);
  printf("\n");
}
int fact(int n)
  int result=1;
  for(int i=1;i<=n;i++)
  {
     result*=i;
  return result;
}
void producePermutation(int n) {
```

```
// To store the current permutation
  int a[n];
  // To store the current directions
  bool dir[n];
  // Storing the elements from 1 to n and
  // printing the first permutation.
  for (int i = 0; i < n; i++) {
     a[i] = i + 1;
     printf("%d ", a[i]);
  }
  printf("\n");
  // Initially all directions are set
  // to RIGHT TO LEFT i.e. 0.
  for (int i = 0; i < n; i++)
     dir[i] = right_to_left;
  // For generating permutations in order.
  for (int i = 1; i < fact(n); i++)
     produceOnePermutation(a, dir, n);
}
void main()
  int n;
  printf("\nEnter the number of objects whose permutations are to be generated: ");
  scanf("%d",&n);
  producePermutation(n);
}
```

Output:

Observation:

```
13-07-23
a) Implement Johnson Trotter algorithm to generate permutations.
#Include (Stalio. hs.
                                                                           for (120; icn; 14
#include (Stol bool, h)
                                                                            it(a[i]>mobile
                                                                               it- (dir [a[i
Hodefine left to right true.
                                                                                  dir [aci]
Halfine right to left labe
                                                                               else it (d
                                                                                 direac
 int getthe of mobile (int all), int n, int mobile) (
    for (int 1=0; icn; 1++) {
                                                                          for (1-0; icn; it
            it (a(i) == mobile)
                                                                              print f(" 40
             return i+1;
                                                                          Beyotten 1 1 1 ):
      return o:
                                                                        jut fact (int n)
                                                                         I but result =
 int getmobile (int all, bool dir [] , int n) of
                                                                             for (1-1; 1 ==
                                                                                 result.
       int mobile prev = 0, mobile = 0;
                                                                             return resul
      for (int i=0; ikn; i+n) {
              i+(diraci)-1)== right-to-lyt 44 11=0)1
                                                                         world produce Permute
                it(ali] > aliB-1] 44 ali]> mobile-prev) ?
                                                                             bool
                       mobile = a[i];
                                                                             for (1-
                  mobile-prev = mobile,
                                                                                  ali
      it (dir Easil-1) == left to right 96 1;=n-1) 1
              it (a(i) > a(i+1) 45 a (i) > mobile-prev) d.
                mobile = a (i];
                mobile prev = mobile;
   it (mobile == 0 dob mobile - prev == 0)
       return 0;
   else
          return mobile;
void produce On Permutation (int a [], bool dir [], int n) f
          int mobile = get Mobile (a, altr, n);
          int pos = ger pos of mobile (a, n, mobile);
    if (dir [a(pos-1)-1] == right_to_left) {
             int temp = a [pos-17;
           a [pos-1) = a [pos-2];
           a (pos-2) = temp;
   elseif (dir [a [pos-1]-1] = = upt-to_sight ).
          int temp = a(pos);
           acpost = a (pos-1);
          a (pos -i) = temp;
```

```
mutation
                     for (1=0; icn; i++) d
                      it (ari) > mobile ) ti
                         1+ (dir[a[1]-D== left-10_ right)
                            ; Hot of the sight to loft;
                        else it (dir[a[i]-1] = = right_to_left.)
                            dir[a[]-]= left_to_right;
                  for (1=0; icn; 1++)
                      printf(" yod", a[i]);
                  beport & C 11 vo 11);
                   fact (int n)
                   int result = 1;
for (1-1; 1 (=n; 1++)
                         result *= 1;
                    return result;
                void produce Permutation (int n) 1.
                    int a(n);
                    bool dir (n);
                    for (1-0; icn; 1+7) d
                         printer yed , alis);
                  for (1=0; icn; ith)
                       direct = right to left;
                   for (1=1) 1 ( fact (n); 1++)
                       produce On Permutation (as dir, 11);
              void maint)
               int n; printer novey objects whose permutation to be generated in"); example and ", 60,77
                  produce Permutation (n);
               Enter the number of objects whose permutations to be generated:3
             output;
                    3
                      2
                3
                    21
                3
                   31
                2
```