Name : Ravi. Sajjanar
USN : 1BM19CS127
Course : ADA Lab CIE-1
Code : 19CS4PCADA
sem/sec : 4-C
Batch : 2nd

Date 10/6/2021
Sign [illegible]

Q7) Implement Johnson Trotter algorithm to generate permutation:

```
#include <stdio.h>
#include <math.h>
int left_to_right=1;
int right_to_left=0;

void swap( int *x ,int *y) {
      int temp= *x;
      *x = *y;
      *y= temp;
   }
int searchchar( int a[], int mobile , int n){
   int i;
   for (i=0; i<n; i++){
      if (a[i]==mobile)
         return i+1;
      }
   }
```

```
int getmobile(int a[], int n, int dir[]) {
    int mobile=0;
    int mobile_prev=0;
    for (int i=0; i<n; i++) {
    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;
          }
      }
   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; }
  }
 int printonprem(int a[], int dir[], int n) {
    int mobile = getmobile(a, n, dir);
     int pos = searchchar (a, mobile, n);
      if (dir[a[pos-1]-1] == right_to_left)
           {
   swap(&a[pos-1], & a[pos-2]);
           }
```

Scanned by CamScanner

```
else {
    swap(&a[pos-1], &a[pos]);
  }
for (int i=0; i<n; i++) {
    if (a[i] >mobile) {
        if (dir [a[i]-1]= right_to_left) {
            dir [a[i]-1]= left-to-right;
            }
        else {
            dir [a[i]-1]= right_to_left;
            }
      }
   }
for (int i=0; i<n; i++) {
        printf ("%d\t", a[i]);
   }
  printf ("\n");
}
Int fact (int n) {
      int p=1;
      for (int i=1; i<=n; i++) {
              p = p*i;
          }
        return p;
      }
```

Scanned by CamScanner

for

```
for (int i=0; i<n; i++) {
    if (a[i]>mobile) {
  if ( dir [a[i]-1] == right_to_left ) {
     dir [a [i] -1 ] = left_to_right;
     }
 else {
    dir[a [i]-1]= right_to_left;
     }
    }
 }
for (int i=0; i<n; i+r){
        print f("%d" ,a[i]);
      }
printf ("\n");
}
int fact (int n) {
        int p=1;
         for (int i=1; j<=n; i++) {
              p = p*i; }
          return p;
      }

void per (int n) {
          int a[n];
          int dir [n];
          for (int i=0; i< n; i++) {
               a[i] = i+1;
          }    printf ("%d" ,a[i]);
```

```
printf("\n");
  for (int i=0; i<n; i++)
  { dir[i]= right_to_left;
   }
for (int i=0; i<(fact(n)-1); i++) {
       printonprem (a, dir, n);
      }
   }
int main() {
   int n;
   printf("Enter the no of terms\n");
    scanf("%d", &n);
     per(n);
    }
```

Scanned by CamScanner

Pg No.

1BM19CS127

## Modification:

Generate permutation for ABCD

```
# include < stdio.h>
# include < math.h>
 int left_to_right = 1;
 int right_to_left = 0;

 void swap (char * x, char *y)
 { char temp = * x;
     * x = * y;
     * y = temp;
   }
 int searchcharr (char a[], int mobile, int n){
      int i;
      for (i=0; i<n; i++){
        if (a[i]==mobile)
          return i+1;
        }
      }
 int getmobile (char a[], int n, int dir[]){
      char mobile = ' ';
      char mobile_prev= '';
      for (int i=0; i<n; i++){
        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;
            }
```

Scanned by CamScanner

```
    }
  }
  if (mobile == 0 && mobile_prev == 0) {
          return 0;
     }
    else { return mobile;
       }
   }

int printonprem (char a[], int dir[], int n) {
      int mobile = getmobile (a, n, dir);
      printf ("%d \t", mobile);
      int pos = searchcharr (a, mobile, n);
      if (dir [a [pos-1] - 1] = = right_to_left) {
              swap (&a [pos-1], &a [pos-2]);
        }
      else { swap (&a [pos-1], &a [pos]);
        }
    for (int i = 0; i<n; i++) {
         if  a[i] > mobile) {
             if (dir [a [i] - 1] = = right_to_left) {
                  dir [a [i] - 1] = left_to_right;
               }
         else { dir [a[i] - 1] = right_to_left;
            }
        }
     }
    for (int i = 0; i<n; i++) {
              printf ("%c\t", a[i]);
       }
    printf (" \n"); }
```

```
int fact (int n) {
        int p=1;
        for (int i=1 ; i<=n; i++) {
                p=p*i;
        }
        return p;
    }
void per (int n) {
        char a[n];
        int dir [n];
        int k='A';
    for (int i=0; i<n; i++, k++) {
            a[i]=k;
            printf("%c\t", a[i]);
        }
    printf("\n");
     for (int i=0; i<n ;i++) {
            dir[i]= right_to_left;
        }
     for (int i=0; i< (fact(n)-1); i++) {
             printonprem (a, dir, n);
         }
    }
 int main() {
          per(4);
      }
```

=