Ravi Sajjanar

1BM19CS127

Program: Implement Johnson Trotter algorithm to generate permutations.
Modification: Generate permutation for ABCD.

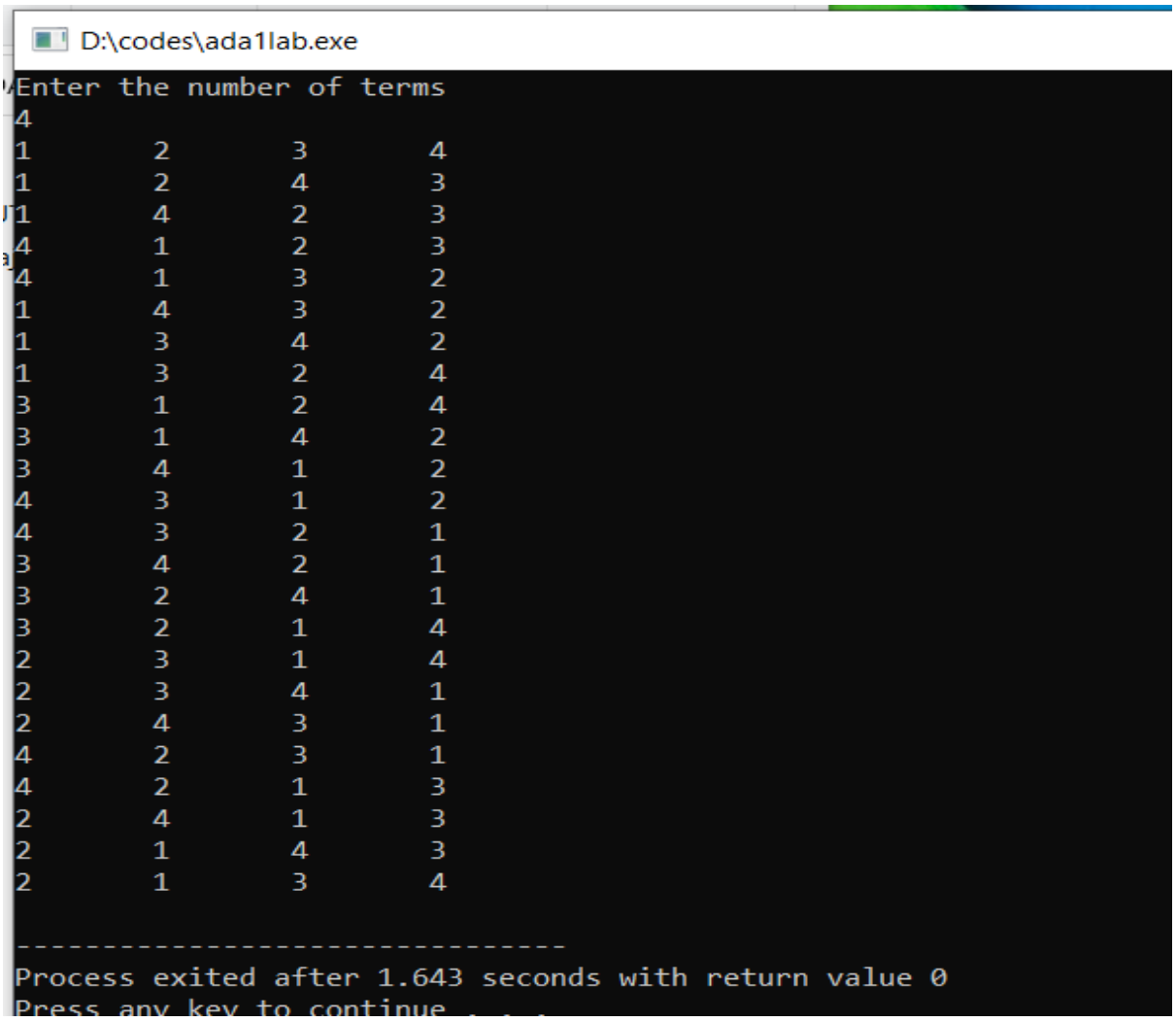
Algo Steps :

1)Generate the initial permutation

2)In the while loop till the mobile component exists

A. Find the largest mobile Component k
B. Swap this mobile component with the immediate adjacent integer (pointed by arrow)
C. Reverse the direction of arrow of all the integers larger than k

OUTPUT :



```
#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 searcharr(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;
```

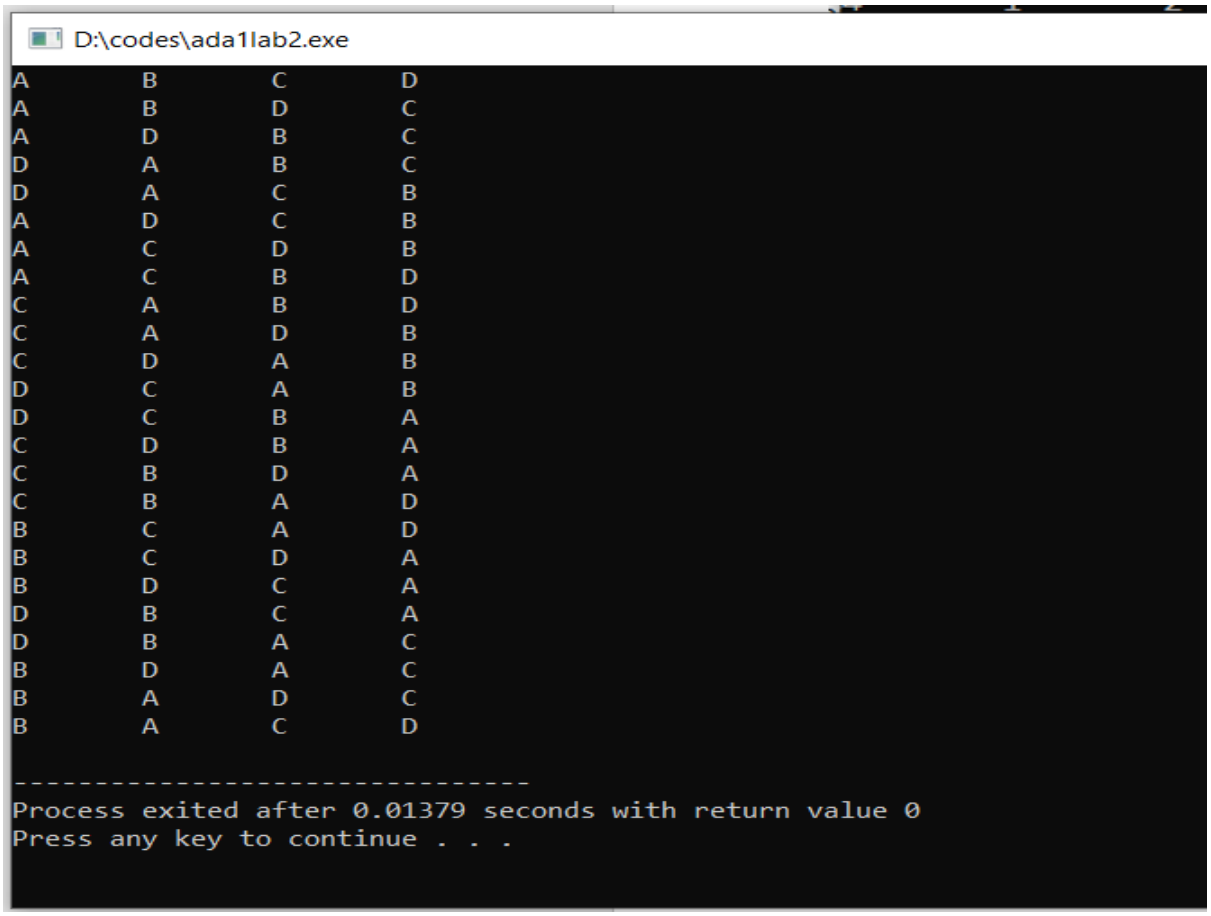
```
        }
      }
      if(dir[a[i]-1]==left_to_right && i!=n-1)
      {
        if(a[i]>a[i+1] && a[i]>mobile_prev)
        {
          mobile=a[i];
          mobile_prev=mobile;
        }

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

```
    printf("%d\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()
{
  int n;
  printf("Enter the number of terms\n");
  scanf("%d",&n);
  per(n);
}
```

OUTPUT :

For modified Program




```
#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 searcharr(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;

      }
    }
    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(char a[],int dir[],int n)
{
  int mobile=getmobile(a,n,dir);
  //printf("%d\t",mobile);
  int pos=searcharr(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);
}
```