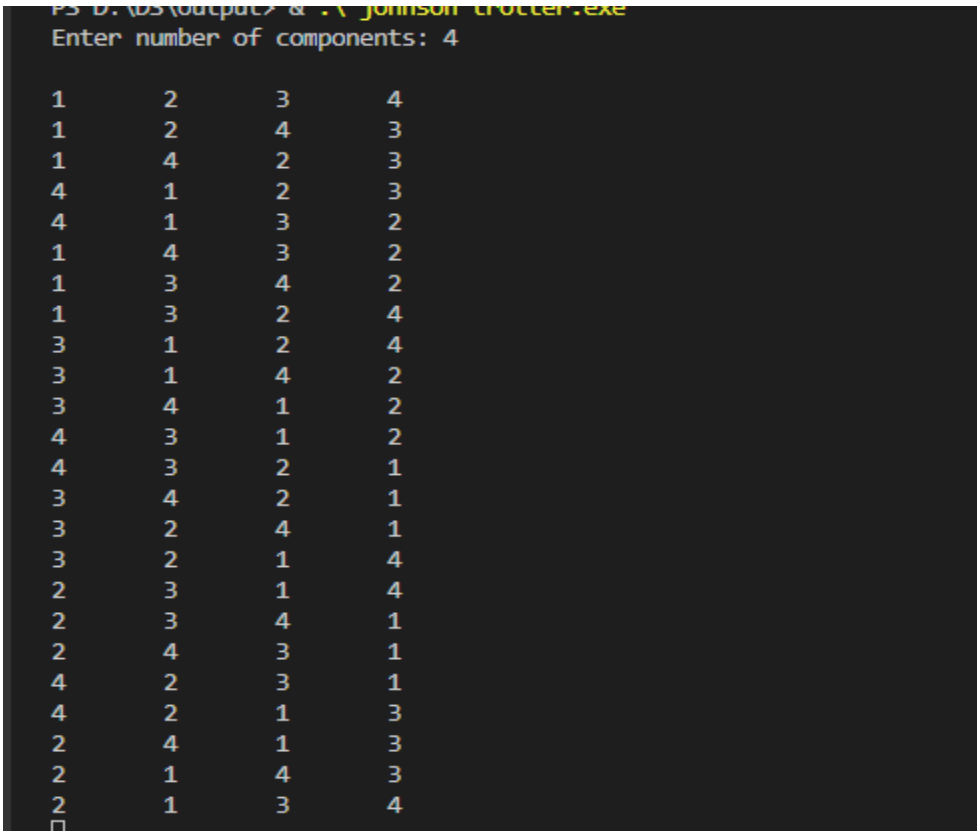
PROGRAM 3

Implement Johnson Trotter algorithm to generate permutations.

```
#include <stdio.h>
int NN, i, count=0;
int p[100], pi[100];
int dir[100];
void PrintPerm(){
  int i;
  for (i=1; i <= NN; ++i)
    printf( "%d\t", p[i] );
}
void PrintTrans( int x, int y ){
  //printf( " (%d %d)", x, y );
  printf( "\n" );
}
void Move( int x, int d ){
  int z;
  PrintTrans( pi[x], pi[x]+d );
  z = p[pi[x]+d];
  p[pi[x]] = z;
  p[pi[x]+d] = x;
  pi[z] = pi[x];
  pi[x] = pi[x]+d;
}
void Perm ( int n ){
  int i;
  if (n > NN)
    PrintPerm();
  else{
    Perm( n+1 );
    for (i=1; i<=n-1; ++i){
      Move( n, dir[n] );
      Perm( n+1 );
    }
```

```
      dir[n] = -dir[n];
    }
}
void main (){
   printf( "Enter number of components: " );
   scanf( "%d", &NN );
   printf( "\n" );
   for (i=1; i<=NN; ++i){
      dir[i] = -1; p[i] = i;
      pi[i] = i;
   }
   Perm ( 1 );
   printf( "\n" );
}
```

OUTPUT:



OBSERVATION:

Johnson Trotter

```
#include <stdio.h>
#include <conio.h>
int NN, i, count = 0;
int p[100], pi[100];
int dir[100];
void PrintPerm()
{
    int i;
    for (i=1; i<=NN; i++)
        printf ("%d", p[i]);
}
void PrintTrans (int x, int y)
{
    printTrans (int x, int y)
    printf ("(%d %d)", x, y);
    printf ("\n");
}
int Move (int x, int d)
{
    int z;
    PrintTrans (pi[x], pi[x]+d);
    z = p[pi[x]+d];
    p[pi[x]+d] = x;
    pi[z] = pi[x];
    pi[x] = pi[x]+d;
}
void Perm (int n)
{
    int i;
    if (n>NN)
        printPerm();
    else
```

```
else
{
    perm(n+1);
    for(i=1; i<=n-1; ++i)
    {
        Move(n, dis[n]);
        Perm(n+1);
    }
    dis[n] = -dis[n];
}
void main()
{
    printf("Enter number of :");
    scanf("%d", &NN);
    printf("\n");
    for(i=1; i<=NN; ++i)
    {
        dis[i] = -1;
        p[i] = i;
        p[i] = i;
    }
    Perm(1);
    printf("\n");
    getch("\n");
}
```

Output:-

Enter the number of components : 4

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

1 4 3 2

1 3 4 2

1 3 2 4

3 1 2 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

3 2 1 4

2 3 1 4

2 3 4 1

2 4 3 1

4 2 3 1

4 2 1 3

2 4 1 3

2 1 4 3

2 1 3 4