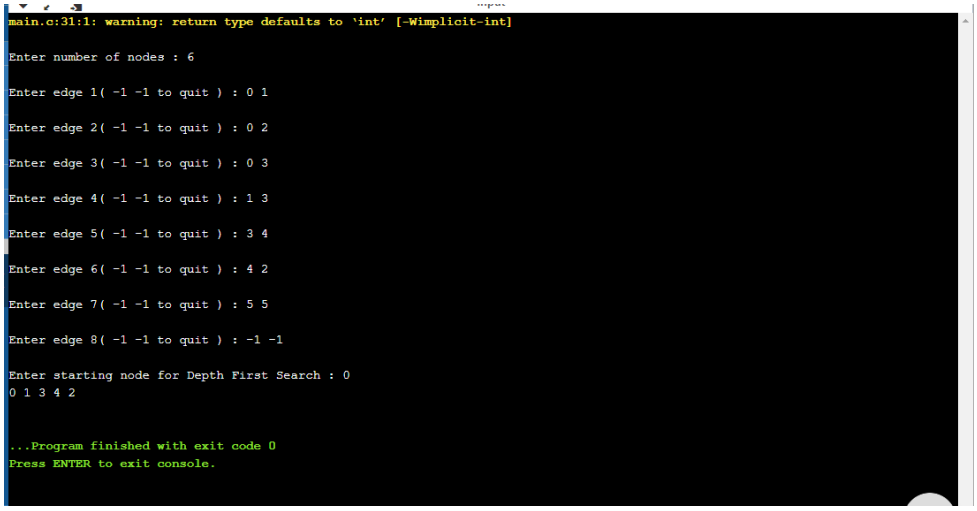
1) Graph traversal technique DFS (using stack)

```
#include<stdio.h>
#include<stdlib.h>

#define MAX 100

#define initial 1
#define visited 2

int n;    /* Number of nodes in the graph */
int adj[MAX][MAX]; /*Adjacency Matrix*/
int state[MAX]; /*Can be initial or visited */

void DF_Traversal();
void DFS(int v);
void create_graph();

int stack[MAX];
int top = -1;
void push(int v);
int pop();
int isEmpty_stack();

main()
{
      create_graph();
      DF_Traversal();
}/*End of main()*/

void DF_Traversal()
{
      int v;

      for(v=0; v<n; v++)
            state[v]=initial;

      printf("\nEnter starting node for Depth First Search : ");
      scanf("%d",&v);
      DFS(v);
      printf("\n");
}/*End of DF_Traversal( )*/

void DFS(int v)
{
      int i;
      push(v);
```

```
        while(!isEmpty_stack())
        {
                v = pop();
                if(state[v]==initial)
                {
                        printf("%d ",v);
                        state[v]=visited;
                }
                for(i=n-1; i>=0; i--)
                {
                        if(adj[v][i]==1 && state[i]==initial)
                                push(i);
                }
        }
}/*End of DFS( )*/

void push(int v)
{
        if(top == (MAX-1))
        {
                printf("\nStack Overflow\n");
                return;
        }
        top=top+1;
        stack[top] = v;

}/*End of push()*/

int pop()
{
        int v;
        if(top == -1)
        {
                printf("\nStack Underflow\n");
                exit(1);
        }
        else
        {
                v = stack[top];
                top=top-1;
                return v;
        }
}/*End of pop()*/

int isEmpty_stack( )
{
  if(top == -1)
```

```
        return 1;
  else
        return 0;
}/*End if isEmpty_stack()*/

void create_graph()
{
      int i,max_edges,origin,destin;

      printf("\nEnter number of nodes : ");
      scanf("%d",&n);
      max_edges=n*(n-1);

      for(i=1;i<=max_edges;i++)
      {
            printf("\nEnter edge %d( -1 -1 to quit ) : ",i);
            scanf("%d %d",&origin,&destin);

            if( (origin == -1) && (destin == -1) )
                  break;

            if( origin >= n || destin >= n || origin<0 || destin<0)
            {
                  printf("\nInvalid edge!\n");
                  i--;
            }
            else
            {
                  adj[origin][destin] = 1;
            }
      }
}
```

Output:


main.c:31:1: warning: return type defaults to 'int' [-Wimplicit-int]
Enter number of nodes : 6
Enter edge 1(-1 -1 to quit) : 0 1
Enter edge 2(-1 -1 to quit) : 0 2
Enter edge 3(-1 -1 to quit) : 0 3
Enter edge 4(-1 -1 to quit) : 1 3
Enter edge 5(-1 -1 to quit) : 3 4
Enter edge 6(-1 -1 to quit) : 4 2
Enter edge 7(-1 -1 to quit) : 5 5
Enter edge 8(-1 -1 to quit) : -1 -1
Enter starting node for Depth First Search : 0
0 1 3 4 2
...Program finished with exit code 0
Press ENTER to exit console.