Data Structure and Algorithm Lab Experiment

Graph Traversal

Name: OM ASHISH MISHRA

Registration Number: 16BCE0789

Slot: B2

The Pseudo code:

- First we declare the header files.
- Then we ask the user for directed and undirected graph.
- We then ask the user for no of vertices.
- If directed graph is chosen then we take the input from user and see to it whether the line is exiting between the 2 vertices.
- According to that we get the in points and out points and are able to draw the graph.
- Same for undirected graph.
- Hence we print the desired graph by using adjacency matrix.

The Code:

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
  int option;
  do
  {
      printf("\n A Program to represent a Graph by using an ");
          printf("Adjacency Matrix method \n ");
          printf("\n 1. Directed Graph ");
          printf("\n 2. Un-Directed Graph ");
          printf("\n 3. Exit ");
          printf("\n\n Select a proper option : ");
          scanf("%d", &option);
          switch(option)
          {
        case 1 : dir_graph();
              break;
        case 2 : undir_graph();
              break;
        case 3 : exit(0);
          } // switch
   }while(1);
}

int dir_graph()
{
```

```
   int adj_mat[50][50];
   int n;
   int in_deg, out_deg, i, j;
   printf("\n How Many Vertices ? : ");
   scanf("%d", &n);
   read_graph(adj_mat, n);
   printf("\n Vertex \t In_Degree \t Out_Degree \t Total_Degree ");
   for (i = 1; i <= n ; i++ )
   {
      in_deg = out_deg = 0;
           for ( j = 1 ; j <= n ; j++ )
           {
         if ( adj_mat[j][i] == 1 )
            in_deg++;
           }
      for ( j = 1 ; j <= n ; j++ )
         if (adj_mat[i][j] == 1 )
            out_deg++;
         printf("\n\n %5d\t\t\t%d\t\t%d\t\t%d\n\n",i,in_deg,out_deg,in_deg+out_deg);
   }
   return;
}

int undir_graph()
{
   int adj_mat[50][50];
   int deg, i, j, n;
   printf("\n How Many Vertices ? : ");
   scanf("%d", &n);
   read_graph(adj_mat, n);
   printf("\n Vertex \t Degree ");
   for ( i = 1 ; i <= n ; i++ )
   {
      deg = 0;
      for ( j = 1 ; j <= n ; j++ )
         if ( adj_mat[i][j] == 1)
            deg++;
      printf("\n\n %5d \t\t %d\n\n", i, deg);
   }
   return;
}

int read_graph ( int adj_mat[50][50], int n )
{
   int i, j;
   char reply;
   for ( i = 1 ; i <= n ; i++ )
   {
```

```
        for ( j = 1 ; j <= n ; j++ )
        {
            if ( i == j )
            {
                adj_mat[i][j] = 0;
                        continue;
            }
            printf("\n Vertices %d & %d are Adjacent ? (Y/N) :",i,j);
            scanf("%c", &reply);
            if ( reply == 'y' || reply == 'Y' )
                adj_mat[i][j] = 1;
            else
                adj_mat[i][j] = 0;
              }
    }
    return;
}
```

The Output:

Directed graph:

```
"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\graph_traversal.exe"

A Program to represent a Graph by using an Adjacency Matrix m

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option : 1

How Many Vertices ? : 5

Vertices 1 & 2 are Adjacent ? (Y/N) :
Vertices 1 & 3 are Adjacent ? (Y/N) :y

Vertices 1 & 4 are Adjacent ? (Y/N) :
Vertices 1 & 5 are Adjacent ? (Y/N) :n

Vertices 2 & 1 are Adjacent ? (Y/N) :
Vertices 2 & 3 are Adjacent ? (Y/N) :n

Vertices 2 & 4 are Adjacent ? (Y/N) :
Vertices 2 & 5 are Adjacent ? (Y/N) :n

Vertices 3 & 1 are Adjacent ? (Y/N) :
Vertices 3 & 2 are Adjacent ? (Y/N) :y

Vertices 3 & 4 are Adjacent ? (Y/N) :
Vertices 3 & 5 are Adjacent ? (Y/N) :y

Vertices 4 & 1 are Adjacent ? (Y/N) :
Vertices 4 & 2 are Adjacent ? (Y/N) :y

Vertices 4 & 3 are Adjacent ? (Y/N) :
Vertices 4 & 5 are Adjacent ? (Y/N) :y

Vertices 5 & 1 are Adjacent ? (Y/N) :
Vertices 5 & 2 are Adjacent ? (Y/N) :y

Vertices 5 & 3 are Adjacent ? (Y/N) :
Vertices 5 & 4 are Adjacent ? (Y/N) :n
```

"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\graph_traversal.exe"

```
Vertex          In_Degree       Out_Degree      Total_Degree

    1                   0               1               1


    2                   3               0               3


    3                   1               2               3


    4                   0               2               2


    5                   2               1               3


A Program to represent a Graph by using an Adjacency Matrix method

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option : _
```

Undirected graph:

```
"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\graph_traversal.exe"
A Program to represent a Graph by using an Adjacency Matrix method

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option : 2

How Many Vertices ? : 5

Vertices 1 & 2 are Adjacent ? (Y/N) :
Vertices 1 & 3 are Adjacent ? (Y/N) :y

Vertices 1 & 4 are Adjacent ? (Y/N) :
Vertices 1 & 5 are Adjacent ? (Y/N) :n

Vertices 2 & 1 are Adjacent ? (Y/N) :
Vertices 2 & 3 are Adjacent ? (Y/N) :n

Vertices 2 & 4 are Adjacent ? (Y/N) :
Vertices 2 & 5 are Adjacent ? (Y/N) :Y

Vertices 3 & 1 are Adjacent ? (Y/N) :
Vertices 3 & 2 are Adjacent ? (Y/N) :N

Vertices 3 & 4 are Adjacent ? (Y/N) :
Vertices 3 & 5 are Adjacent ? (Y/N) :N

Vertices 4 & 1 are Adjacent ? (Y/N) :
Vertices 4 & 2 are Adjacent ? (Y/N) :Y

Vertices 4 & 3 are Adjacent ? (Y/N) :
Vertices 4 & 5 are Adjacent ? (Y/N) :N

Vertices 5 & 1 are Adjacent ? (Y/N) :
Vertices 5 & 2 are Adjacent ? (Y/N) :Y

Vertices 5 & 3 are Adjacent ? (Y/N) :
Vertices 5 & 4 are Adjacent ? (Y/N) :N
```

"C:\Users\OM\(OM)\2nd semester\CODE-BLOCKS\graph_traversal.exe"

```
Vertex          Degree

    1           1


    2           1


    3           0


    4           1


    5           1


A Program to represent a Graph by using an Adjacency Matrix method

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option : _
```

THANK YOU