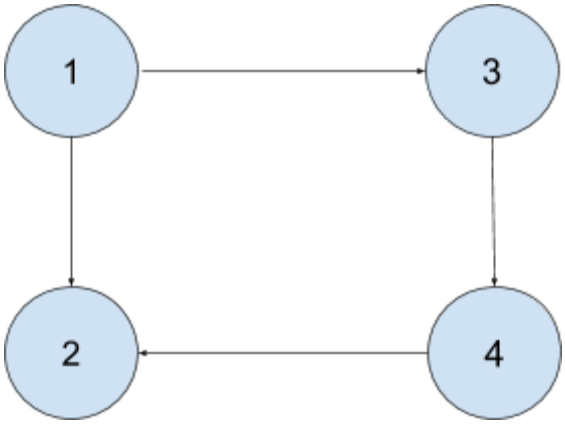
KARIKARAN.V
E/16/172

CO322 – LAB 05

1. Find out what is the Transitive Closure of a graph.

Transitive Closure it the reachability matrix to reach from vertex u to vertex v of a graph. One graph is given, we have to find a vertex v which is reachable from another vertex u,for all pairs of vertices (u, v).The final matrix is the Boolean type. When there is a value 1 for vertex u to vertex v, it means there is at least one path from u to v.

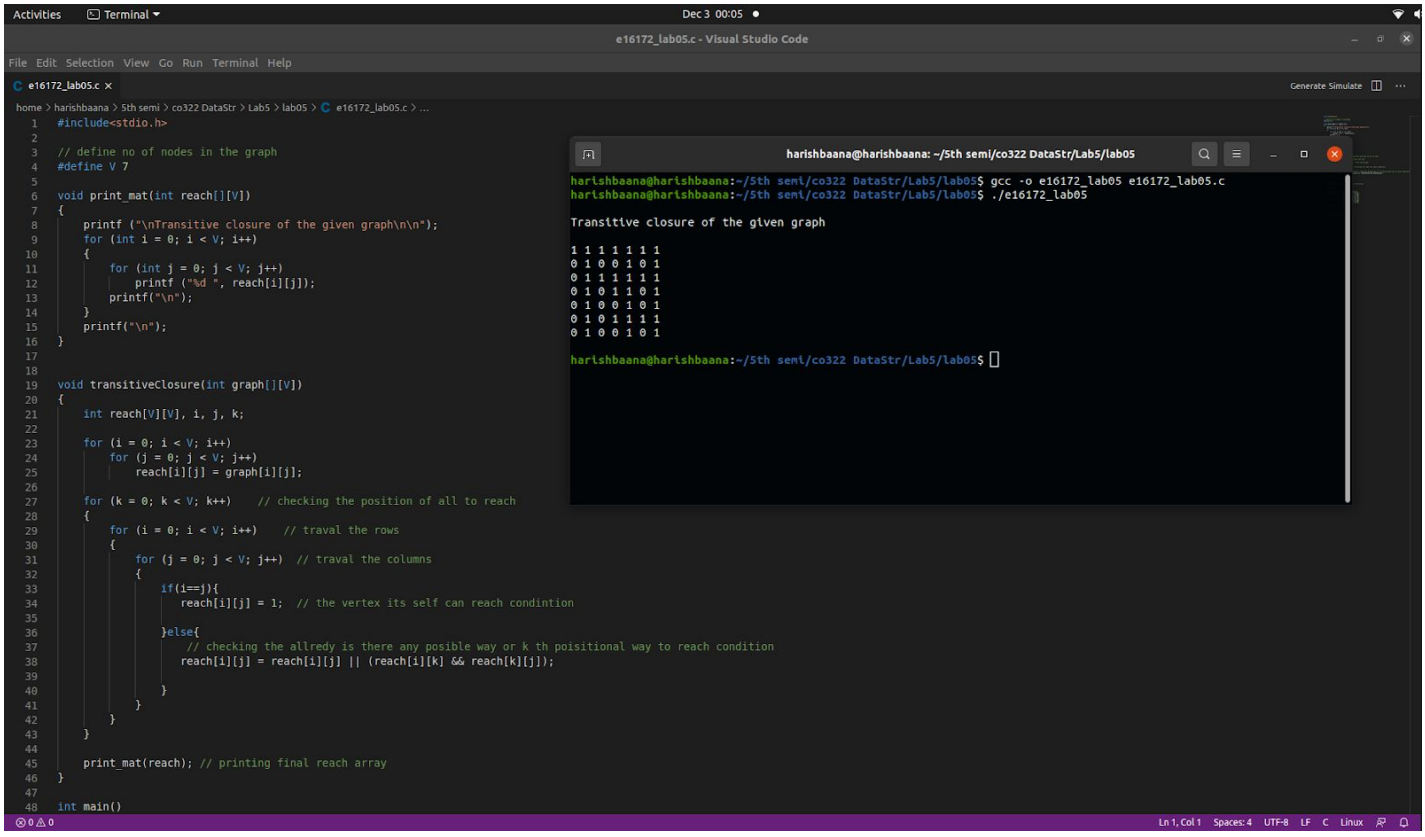
2. Manually compute the Transitive Closure for the following graph:



Transitive Closure matrix for the above graph:

1	1	1	1
0	1	0	0
0	1	1	1
0	1	0	1

3. Based on the Graph Traversal algorithm discussed in the class, write a C program to compute and print the Transitive Closure of a given graph. Use the following graph to test your program:




```c
#include<stdio.h>
 // define no of nodes in the graph
#define V 7
 void print_mat(int reach[][V])
{
   printf ("\nTransitive closure of the given graph\n\n");
   for (int i = 0; i < V; i++)
   {
        for (int j = 0; j < V; j++)
            printf ("%d ", reach[i][j]);
        printf("\n");
```

```
   }
   printf("\n");
}

 void transitiveClosure(int graph[][V])
{
   int reach[V][V], i, j, k;
    for (i = 0; i < V; i++)
       for (j = 0; j < V; j++)
           reach[i][j] = graph[i][j];
    for (k = 0; k < V; k++)     // checking the position of all to reach
   {
       for (i = 0; i < V; i++)     // traval the rows
       {
           for (j = 0; j < V; j++)  // traval the columns
           {
               if(i==j){
                  reach[i][j] = 1;  // the vertex its self can reach

               }else{


                  reach[i][j] = reach[i][j] || (reach[i][k] && reach[k][j]);

               }
           }
       }
   }
    print_mat(reach); // printing final reach array
}

 int main()
{
    // Insert adjacency matrix of the graph
   int graph[V][V] = { {0, 1, 1, 0, 0, 0, 0},
                       {0, 0, 0, 0, 1, 0, 0},
                       {0, 0, 0, 0, 0, 1, 0},
                       {0, 1, 0, 0, 1, 0, 1},
                       {0, 0, 0, 0, 0, 0, 1},
                       {0, 0, 0, 1, 0, 0, 0},
                       {0, 1, 0, 0, 0, 0, 0}
```

```
                    };

    // Print the solution
    transitiveClosure(graph);
    return 0;
}
```