Q1: - Create a program to implement Graphs with Adjacency matrix.

Input:-

Console.WriteLine("Welcome \tTask No 1");

Console.WriteLine("Enter the number of Nodes : ");

int n = int.Parse(Console.ReadLine());

string[] arr = new string[n];

for (int i = 0; i < n; i++)

{

Console.WriteLine((i + 1) + " is connected to nodes?");

arr[i] = Console.ReadLine();

}

int[,] array = new int[n, n];

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

string temp = "" + (j + 1);

if (arr[i].Contains(temp))

{

array[i, j] = 1;

}

else

{

array[i, j] = 0;

}

}

}

Console.Write(" ");

for (int i = 0; i < n; i++)

{

Console.Write(i + 1 + " ");

}

Console.WriteLine();

for (int i = 0; i < array.GetLength(0); i++)

{

Console.Write((i + 1) + " ");

for (int j = 0; j < array.GetLength(1); j++)

{

Console.Write(array[i, j] + " ");

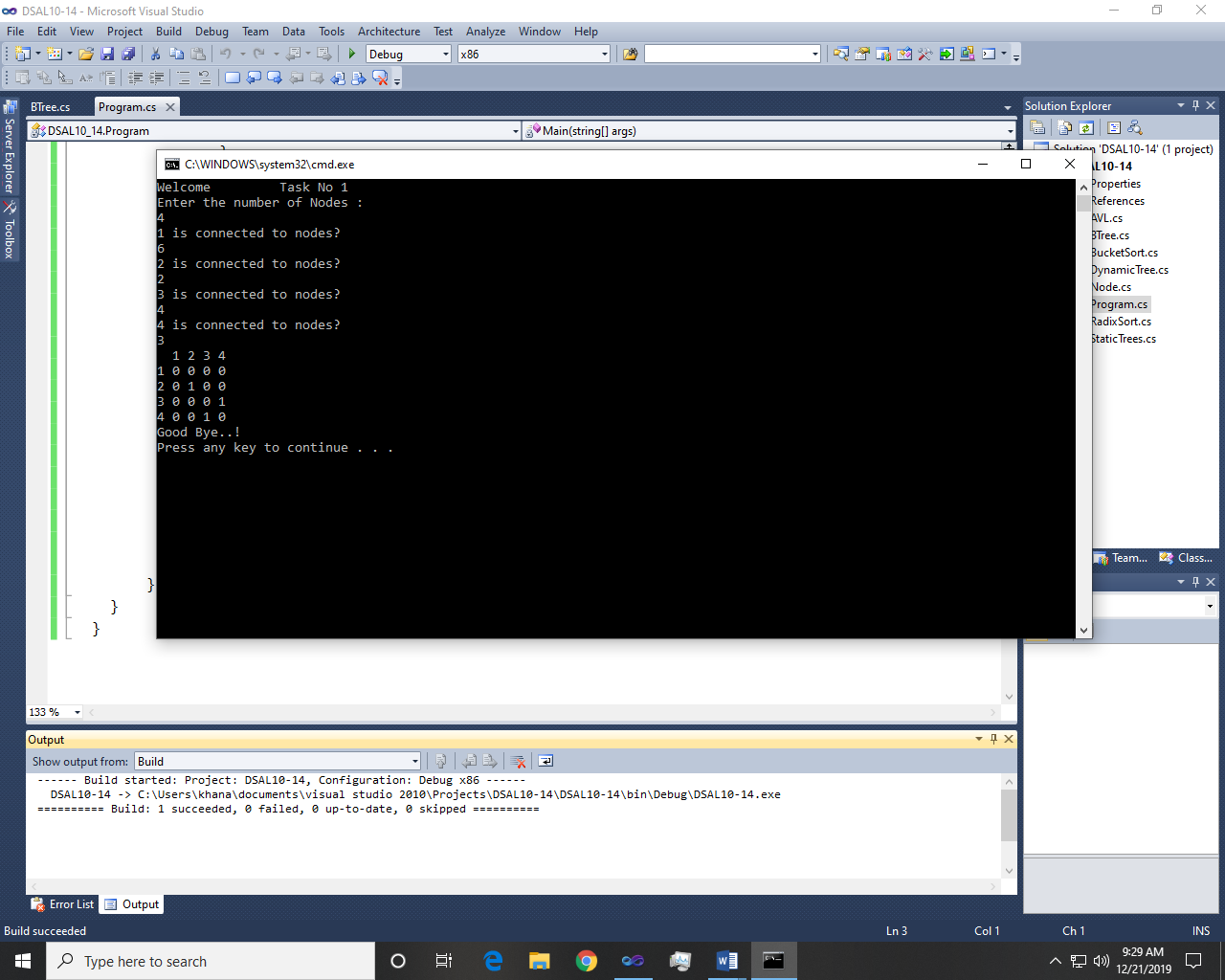
}

Console.WriteLine();

}

Console.WriteLine("Good Bye..!");

Output:-



Q2: - Create a program to implement Graphs with Adjacency list.

Input:-

Console.WriteLine("Welcome \tTask No 2");

Console.WriteLine("Enter the number of nodes");

int n = int.Parse(Console.ReadLine());

string[] arr = new string[n];

for (int i = 0; i < n; i++)

{

Console.WriteLine((i + 1) + " is connected to nodes?");

arr[i] = Console.ReadLine();

}

List<int>[] obj = new List<int>[n];

for (int i = 0; i < n; i++)

{

obj[i] = new List<int>();

string[] temp = arr[i].Split(',');

for (int a = 0; a < temp.Length; a++)

{

obj[i].Add(int.Parse(temp[a]));

}

}

for (int i = 0; i < n; i++)

{

Console.Write((i + 1) + " => ");

for (int l = 0; l < obj[i].Count; l++)

{

Console.Write(obj[i][l] + " ");

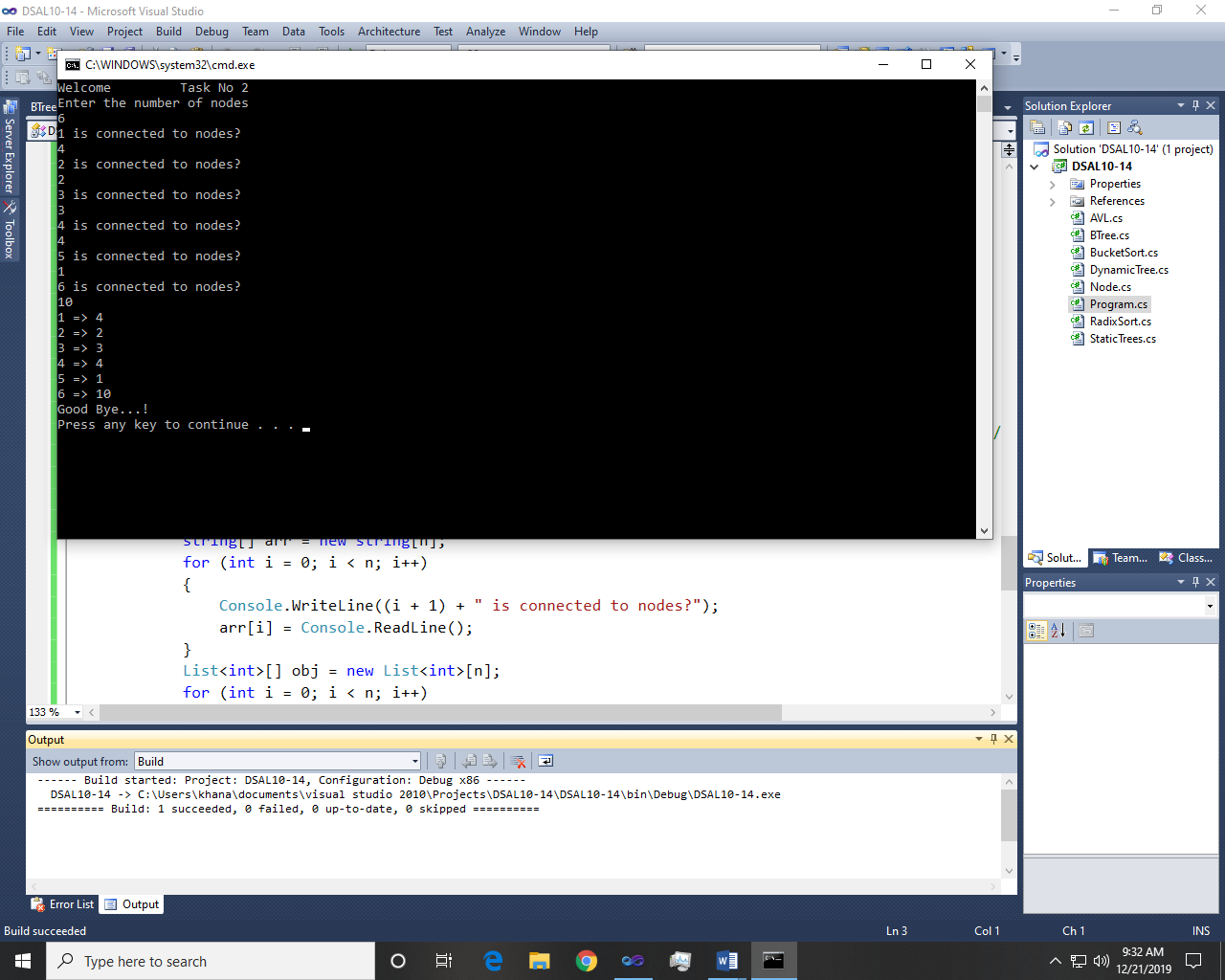
}

Console.WriteLine();

}

Console.WriteLine("Good Bye...!");

Output:-



Q3: - Create a program to implement BFS.

Input:-

Graph:-

class Graph1

{

internal int Vertices;

//adjency list array for all vertices.

internal List<Int32>[] adj;

/\* Example : vertices=4

\* 0->[1,2]

\* 1->[2]

\* 2->[0,3]

\* 3->[]

\*/

//constructor

public Graph1(int v)

{

Vertices = v;

adj = new List<Int32>[v];

//Instantiate adjacecny list for all vertices

for (int i = 0; i < v; i++)

{

adj[i] = new List<Int32>();

}

}

//Add edge from v->w

public void AddEdge(int v, int w)

{

adj[v].Add(w);

}

//Print BFS traversal

//s-> start node

//BFS uses queue as a base.

public void BFS(int s)

{

bool[] visited = new bool[Vertices];

//create queue for BFS

Queue<int> queue = new Queue<int>();

visited[s] = true;

queue.Enqueue(s);

//loop through all nodes in queue

while (queue.Count != 0)

{

//Deque a vertex from queue and print it.

s = queue.Dequeue();

Console.WriteLine("next->" + s);

//Get all adjacent vertices of s

foreach (Int32 next in adj[s])

{

if (!visited[next])

{

visited[next] = true;

queue.Enqueue(next);

}

}

}

}

//DFS traversal

// DFS uses stack as a base.

public void DFS(int s)

{

bool[] visited = new bool[Vertices];

//For DFS use stack

Stack<int> stack = new Stack<int>();

visited[s] = true;

stack.Push(s);

while (stack.Count != 0)

{

s = stack.Pop();

Console.WriteLine("next->" + s);

foreach (int i in adj[s])

{

if (!visited[i])

{

visited[i] = true;

stack.Push(i);

}

}

}

}

public void PrintAdjacecnyMatrix()

{

for (int i = 0; i < Vertices; i++)

{

Console.Write(i + ":[");

string s = "";

foreach (var k in adj[i])

{

s = s + (k + ",");

}

s = s.Substring(0, s.Length - 1);

s = s + "]";

Console.Write(s);

Console.WriteLine();

}

}

Main:-

Console.WriteLine("Welcome \tTask No 3\n");

Graph1 obj = new Graph1(4);

obj.AddEdge(0, 1);

obj.AddEdge(0, 2);

obj.AddEdge(1, 2);

obj.AddEdge(2, 0);

obj.AddEdge(2, 3);

obj.AddEdge(3, 3);

//Print adjacency matrix

obj.PrintAdjacecnyMatrix();

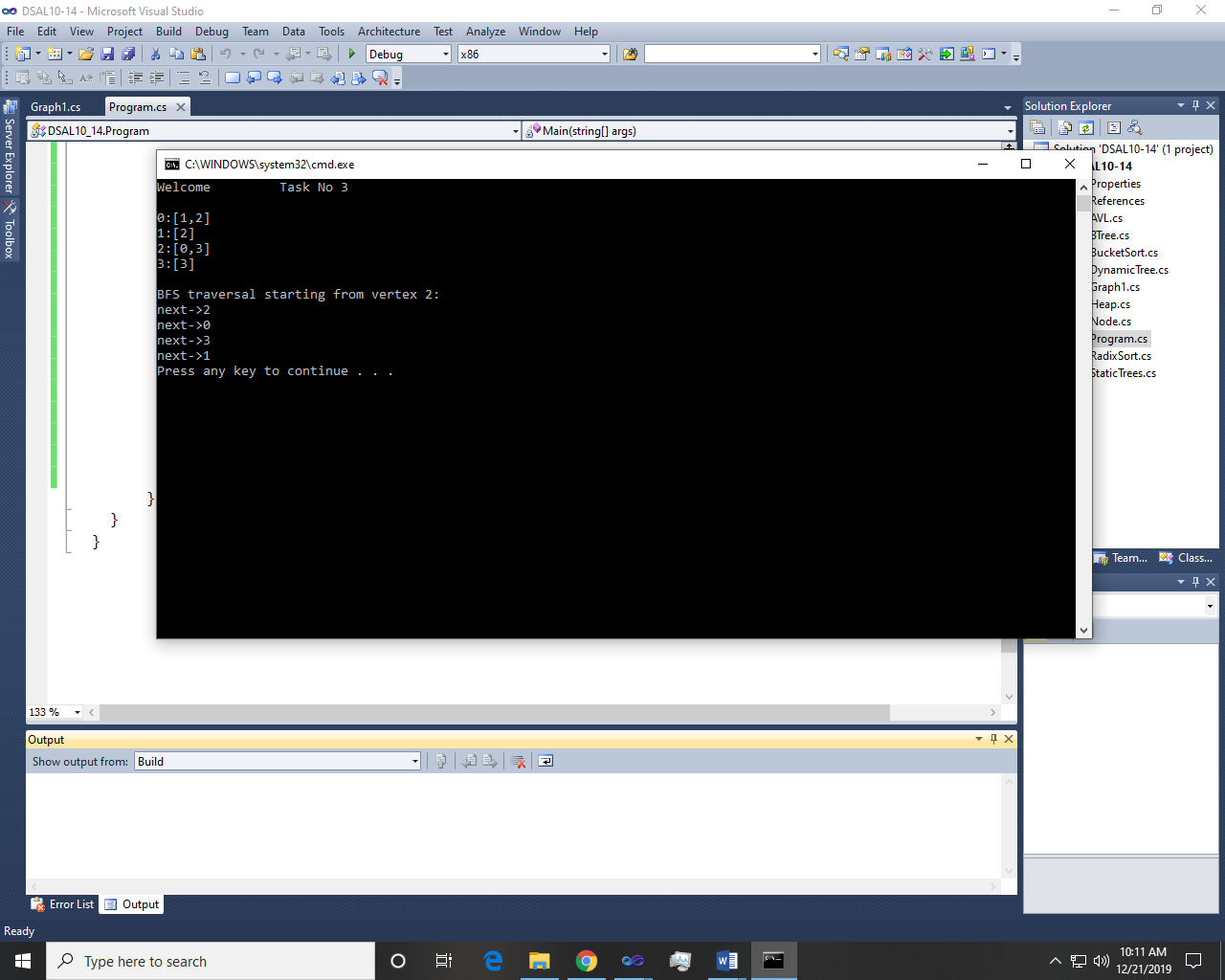
Console.WriteLine("\nBFS traversal starting from vertex 2:");

obj.BFS(2);

//Console.WriteLine("DFS traversal starting from vertex 2:");

//obj.DFS(2);

Output:-



Q4: - Implement heap sort using graph.

Input:-

Heap:-

class Heap

{

int[] r = { 20, 50, 10, 1, 6, 90, 3, 47, 44, 58 };

public void hsort()

{

int i, t;

for (i = 5; i >= 0; i--)

{

adjust(i, 9);

}

for (i = 8; i >= 0; i--)

{

t = r[i + 1];

r[i + 1] = r[0];

r[0] = t;

adjust(0, i);

}

}

private void adjust(int i, int n)

{

int t, j;

try

{

t = r[i];

j = 2 \* i;

while (j <= n)

{

if (j < n && r[j] < r[j + 1])

j++;

if (t >= r[j])

break;

r[j / 2] = r[j];

j \*= 2;

}

r[j / 2] = t;

}

catch (IndexOutOfRangeException e)

{

Console.WriteLine("Array Out of Bounds ", e);

}

}

public void print()

{

for (int i = 0; i < 10; i++)

{

Console.Write("{0}", r[i]+" ");

}

Console.WriteLine("\n");

}

}

Main:-

Console.WriteLine("Welcome \tTask No 4");

Heap obj = new Heap();

Console.WriteLine("Elements Before sorting : ");

obj.print();

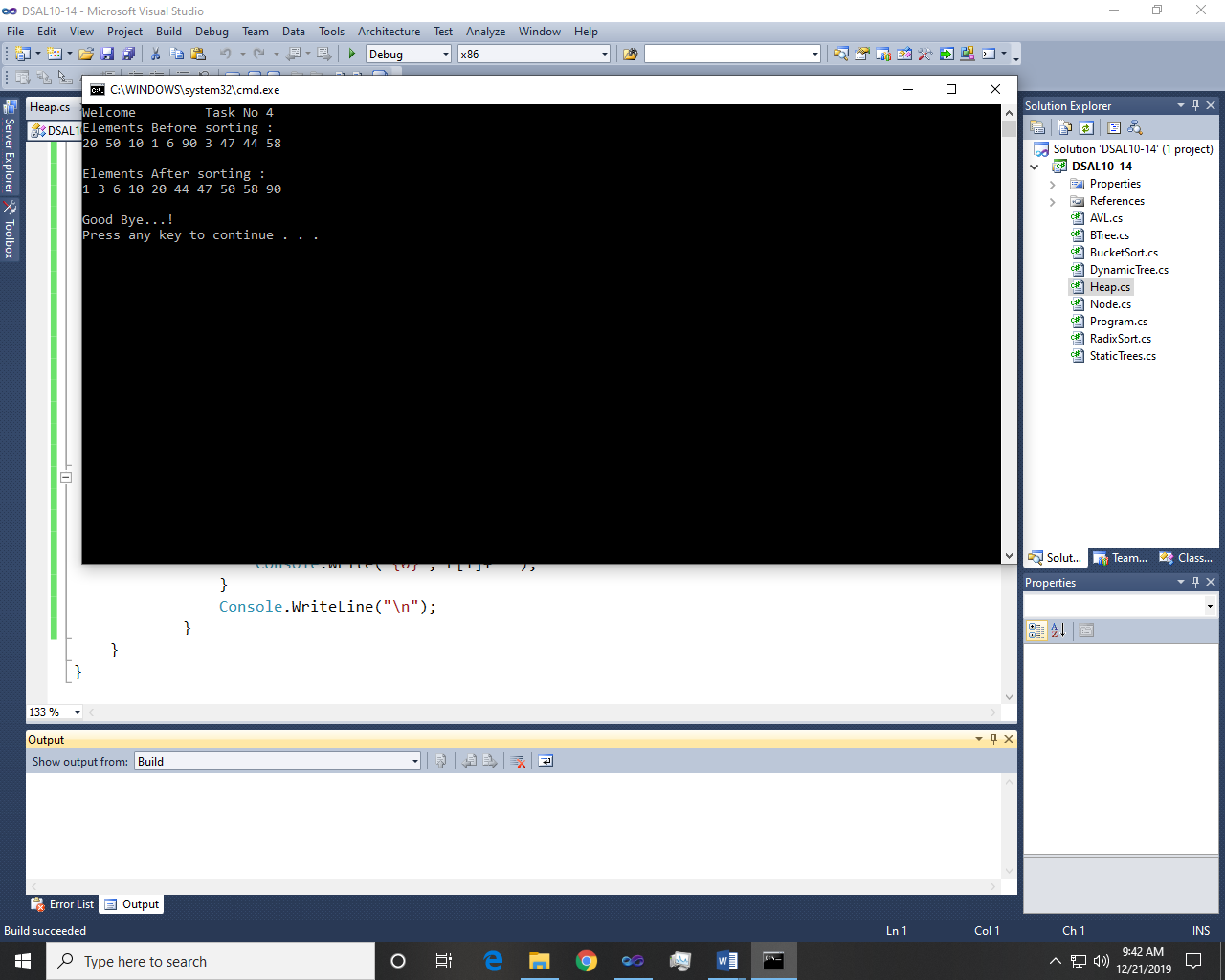
obj.hsort();

Console.WriteLine("Elements After sorting : ");

obj.print();

Console.WriteLine("Good Bye...!");

Output:-



~~~~~~\*\*/**THE END**/\*\*~~~~~~