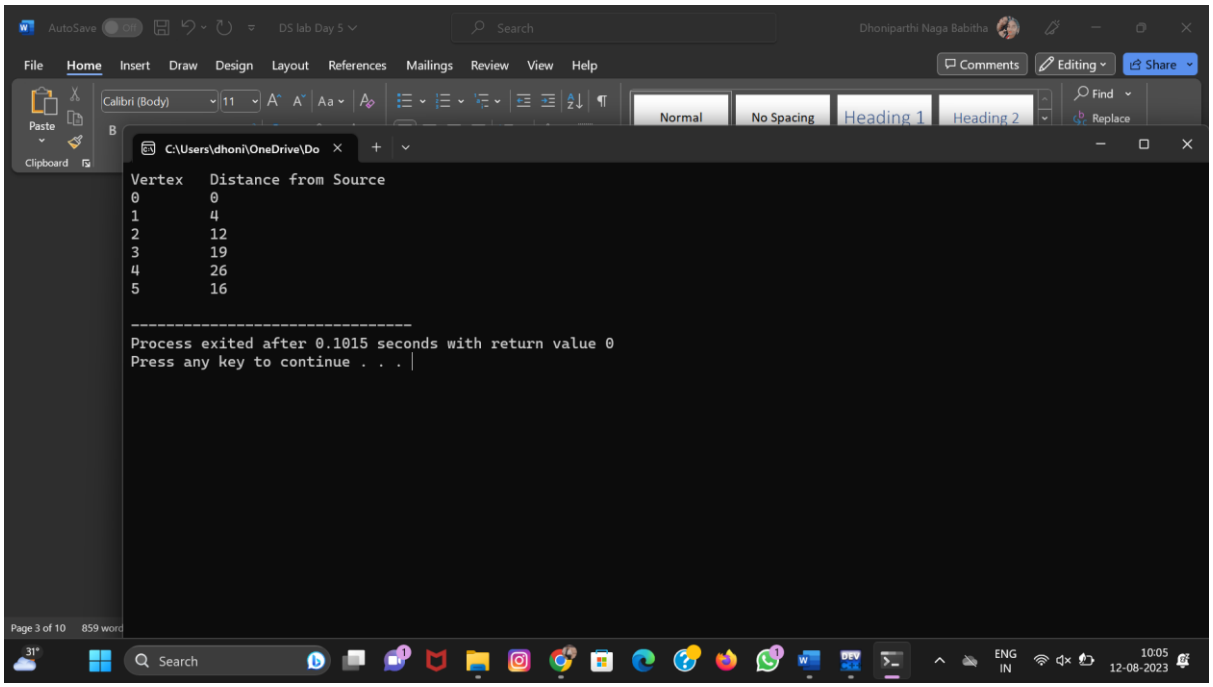
C PROGRAMMING – DATA STRUCTURE - 5

1.Shortest path prims in c programming:

```
#include <stdio.h>
#include <limits.h>
#define V 6 // Number of vertices
int minDistance(int dist[], int visited[]) {
   int min = INT_MAX, min_index;
   for (int v = 0; v < V; v++)
      if (!visited[v] && dist[v] <= min)
         min = dist[v], min_index = v;
   return min_index;
}
void printSolution(int dist[]) {
   printf("Vertex \t Distance from Source\n");
   for (int i = 0; i < V; i++)
      printf("%d \t %d\n", i, dist[i]);
}
void dijkstra(int graph[V][V], int src) {
   int dist[V];
   int visited[V];
   for (int i = 0; i < V; i++) {
      dist[i] = INT_MAX;
      visited[i] = 0;
   }
   dist[src] = 0;
      for (int count = 0; count < V - 1; count++) {
      int u = minDistance(dist, visited);
      visited[u] = 1;
      for (int v = 0; v < V; v++)
         if (!visited[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
```

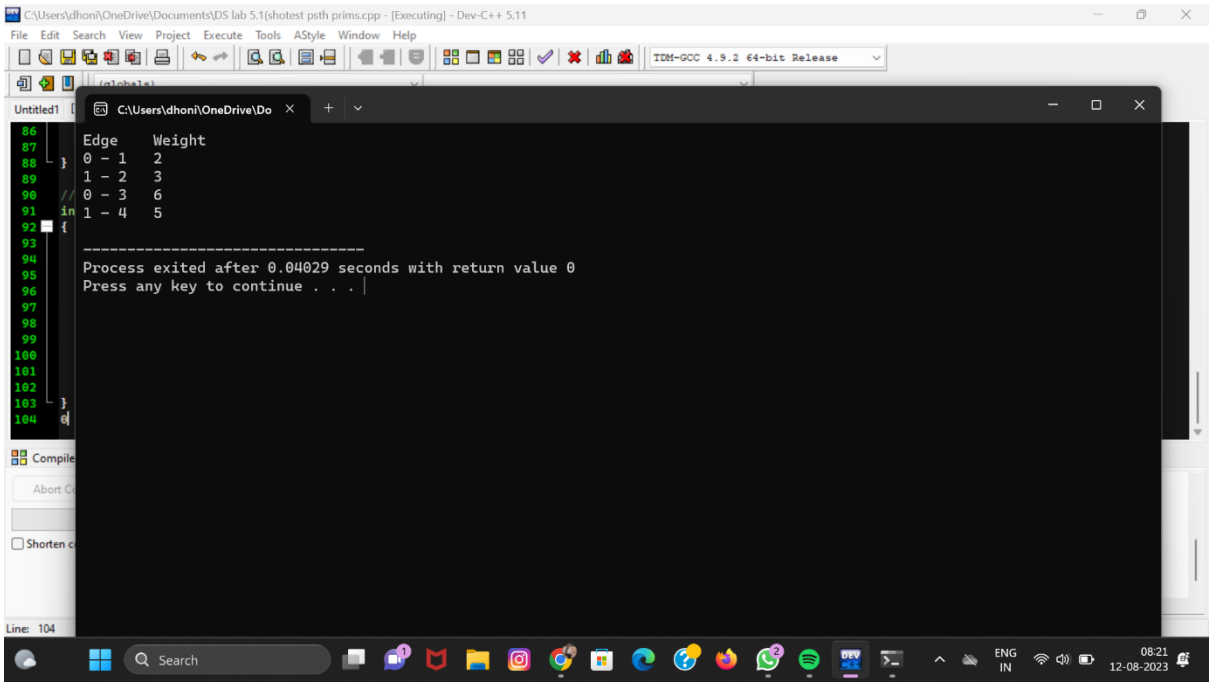
```
            dist[v] = dist[u] + graph[u][v];
    }
    printSolution(dist);
}
int main() {
    int graph[V][V] = {
        {0, 4, 0, 0, 0, 0},
        {4, 0, 8, 0, 0, 0},
        {0, 8, 0, 7, 0, 4},
        {0, 0, 7, 0, 9, 14},
        {0, 0, 0, 9, 0, 10},
        {0, 0, 4, 14, 10, 0}
    }
    dijkstra(graph, 0)
    return 0;
}
```



2.Minimum spanning tree:

```
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
#define V 5
int minKey(int key[], bool mstSet[])
{
	int min = INT_MAX, min_index;
	for (int v = 0; v < V; v++)
		if (mstSet[v] == false && key[v] < min)
			min = key[v], min_index = v;
	return min_index;
}
int printMST(int parent[], int graph[V][V])
{
	printf("Edge \tWeight\n");
	for (int i = 1; i < V; i++)
		printf("%d - %d \t%d \n", parent[i], i,
			graph[i][parent[i]]);
}
void primMST(int graph[V][V])
{
	int parent[V];
	int key[V];
	bool mstSet[V];
	for (int i = 0; i < V; i++)
		key[i] = INT_MAX, mstSet[i] = false;
	key[0] = 0;
```

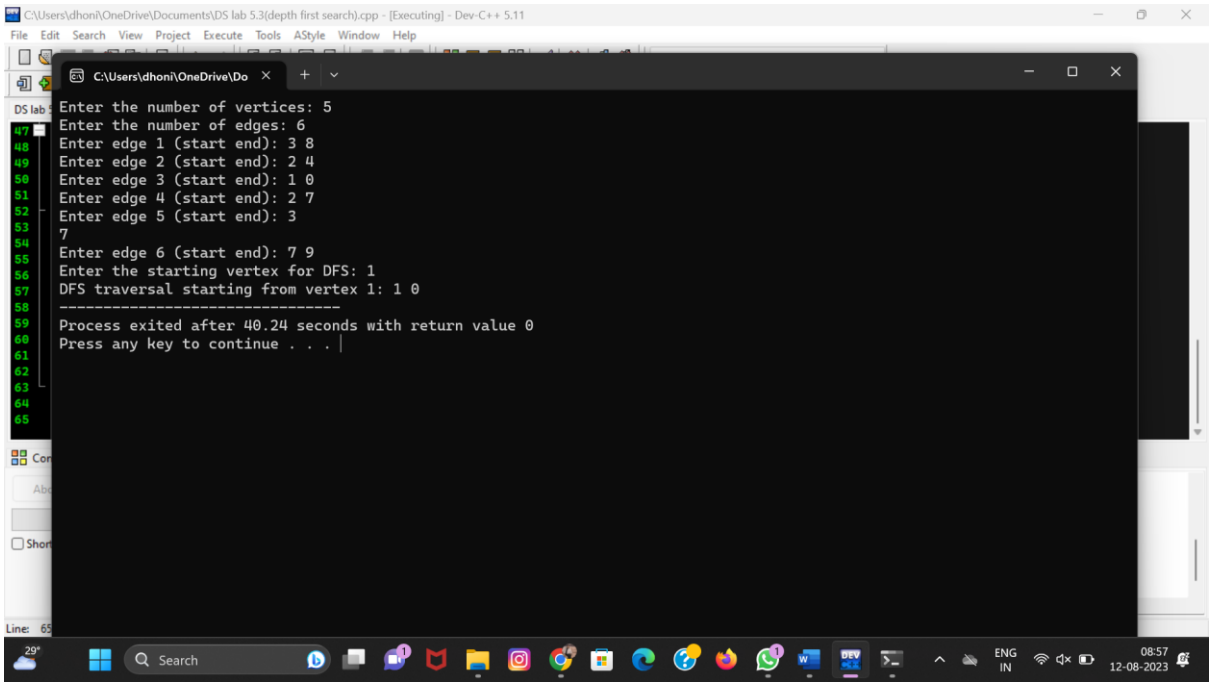
```
	parent[0] = -1;
	for (int count = 0; count < V - 1; count++) {
		int u = minKey(key, mstSet);
		mstSet[u] = true;
		for (int v = 0; v < V; v++)
			if (graph[u][v] && mstSet[v] == false
				&& graph[u][v] < key[v])
				parent[v] = u, key[v] = graph[u][v];
	}

	// print the constructed MST
	printMST(parent, graph);
}

// Driver's code
int main()
{
	int graph[V][V] = {		{ 0, 2, 0, 6, 0 },
				{ 2, 0, 3, 8, 5 },
				{ 0, 3, 0, 0, 7 },
				{ 6, 8, 0, 0, 9 },
				{ 0, 5, 7, 9, 0 } };
	primMST(graph);
	return 0;
}
```



3.Graph transversal depth first search:

```
#include <stdio.h>

#include <stdbool.h>

#define MAX_VERTICES 100

bool visited[MAX_VERTICES];

int adjacencyMatrix[MAX_VERTICES][MAX_VERTICES];

int numVertices;

void initialize() {

   for (int i = 0; i < MAX_VERTICES; i++) {

      visited[i] = false;

      for (int j = 0; j < MAX_VERTICES; j++) {

         adjacencyMatrix[i][j] = 0;

      }

   }

}

void addEdge(int start, int end) {

   adjacencyMatrix[start][end] = 1;

   adjacencyMatrix[end][start] = 1;

}
```

```
void DFS(int vertex) {
    visited[vertex] = true;
    printf("%d ", vertex)
    for (int i = 0; i < numVertices; i++) {
        if (adjacencyMatrix[vertex][i] && !visited[i]) {
            DFS(i);
        }
    }
}
int main() {
    initialize();
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);
    int numEdges;
    printf("Enter the number of edges: ");
    scanf("%d", &numEdges)
    for (int i = 0; i < numEdges; i++) {
        int start, end;
        printf("Enter edge %d (start end): ", i + 1);
        scanf("%d %d", &start, &end);
        addEdge(start, end);
    }
    int startVertex;
    printf("Enter the starting vertex for DFS: ");
    scanf("%d", &startVertex);
    printf("DFS traversal starting from vertex %d: ", startVertex);
    DFS(startVertex)
    return 0;
}
```

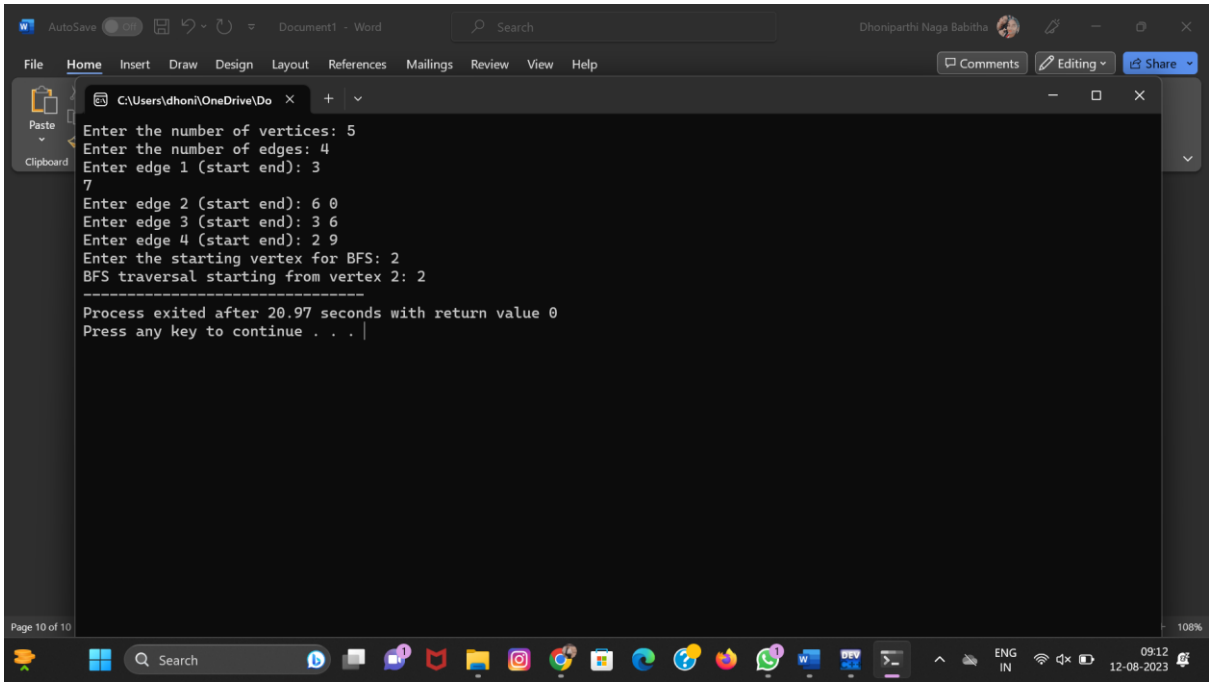


4. Graph transversal [BFS]:

```
#include <stdio.h>

#include <stdbool.h>

#define MAX_VERTICES 100

#define QUEUE_SIZE 100

bool visited[MAX_VERTICES];

int adjacencyMatrix[MAX_VERTICES][MAX_VERTICES];

int numVertices;

int queue[QUEUE_SIZE];

int front = -1, rear = -1;

void initialize() {

   for (int i = 0; i < MAX_VERTICES; i++) {

      visited[i] = false;

      for (int j = 0; j < MAX_VERTICES; j++) {

         adjacencyMatrix[i][j] = 0;

      }

   }

}
```

```
void addEdge(int start, int end) {
    adjacencyMatrix[start][end] = 1;
    adjacencyMatrix[end][start] = 1;
}
void enqueue(int vertex) {
    if (rear == QUEUE_SIZE - 1) {
        printf("Queue is full.\n");
        return;
    }
    if (front == -1)
        front = 0;
    rear++;
    queue[rear] = vertex;
}
int dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return -1;
    }
    int vertex = queue[front];
    front++;
    return vertex;
}

void BFS(int startVertex) {
    visited[startVertex] = true;
    enqueue(startVertex);
    while (front <= rear) {
        int currentVertex = dequeue();
        printf("%d ", currentVertex);
```

```
        for (int i = 0; i < numVertices; i++) {
            if (adjacencyMatrix[currentVertex][i] && !visited[i]) {
                visited[i] = true;
                enqueue(i);
            }
        }
    }
}
int main() {
    initialize();
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);
    int numEdges;
    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);
    for (int i = 0; i < numEdges; i++) {
        int start, end;
        printf("Enter edge %d (start end): ", i + 1);
        scanf("%d %d", &start, &end);
        addEdge(start, end);
    int startVertex;
    printf("Enter the starting vertex for BFS: ");
    scanf("%d", &startVertex);
    printf("BFS traversal starting from vertex %d: ", startVertex);
    BFS(startVertex);
    return 0;
}
```


Document1 - Word
Search
Dhoniparthi Naga Babitha
File
Home
Insert
Draw
Design
Layout
References
Mailings
Review
View
Help
Comments
Editing
Share
Paste
Clipboard
C:\Users\dhoni\OneDrive\Do
Enter the number of vertices: 5
Enter the number of edges: 4
Enter edge 1 (start end): 3
7
Enter edge 2 (start end): 6 0
Enter edge 3 (start end): 3 6
Enter edge 4 (start end): 2 9
Enter the starting vertex for BFS: 2
BFS traversal starting from vertex 2: 2

Process exited after 20.97 seconds with return value 0
Press any key to continue . . .
Page 10 of 10
108%
Search
ENG
IN
09:12
12-08-2023