Input Code :

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

// Define maximum number of vertices in the graph
#define MAX_VERTICES 6

// Adjacency matrix representation of the graph
int graph[MAX_VERTICES][MAX_VERTICES] = {
    {0, 7, 9, 0, 0, 14},
    {7, 0, 10, 15, 0, 0},
    {9, 10, 0, 11, 0, 2},
    {0, 15, 11, 0, 6, 0},
    {0, 0, 0, 6, 0, 9},
    {14, 0, 2, 0, 9, 0}
};

// Depth First Search (DFS)
void DFS(int graph[MAX_VERTICES][MAX_VERTICES], int visited[MAX_VERTICES], int vertex) {
    visited[vertex] = 1;
    printf("%d ", vertex);

    // Explore each adjacent vertex
    for (int i = 0; i < MAX_VERTICES; i++) {
        if (graph[vertex][i] != 0 && !visited[i]) {
            DFS(graph, visited, i);
        }
    }
}

// Breadth First Search (BFS)
void BFS(int graph[MAX_VERTICES][MAX_VERTICES], int visited[MAX_VERTICES], int startVertex) {
    int queue[MAX_VERTICES], front = -1, rear = -1;
    visited[startVertex] = 1;
    queue[++rear] = startVertex;

    while (front != rear) {
        int vertex = queue[++front];
        printf("%d ", vertex);

        // Explore all adjacent vertices of the current vertex
        for (int i = 0; i < MAX_VERTICES; i++) {
            if (graph[vertex][i] != 0 && !visited[i]) {
                visited[i] = 1;
                queue[++rear] = i;
            }
        }
    }
}

// Dijkstra's Algorithm to find the shortest path
void Dijkstra(int graph[MAX_VERTICES][MAX_VERTICES], int startVertex) {
    int dist[MAX_VERTICES], visited[MAX_VERTICES];

    // Initialize distances and visited status
    for (int i = 0; i < MAX_VERTICES; i++) {
        dist[i] = INT_MAX;
        visited[i] = 0;
    }

    dist[startVertex] = 0;

    // Dijkstra's Algorithm main loop
    for (int count = 0; count < MAX_VERTICES - 1; count++) {
        int minDist = INT_MAX, u;

        // Find the vertex with the minimum distance
        for (int v = 0; v < MAX_VERTICES; v++) {
            if (!visited[v] && dist[v] < minDist) {
                minDist = dist[v];
                u = v;
            }
        }

        // Mark the selected vertex as visited
        visited[u] = 1;

        // Update the distance of the adjacent vertices
        for (int v = 0; v < MAX_VERTICES; v++) {
            if (graph[u][v] != 0 && !visited[v] && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }

    // Print the shortest distances
    printf("Shortest distances from vertex %d:\n", startVertex);
    for (int i = 0; i < MAX_VERTICES; i++) {
        printf("To %d: %d\n", i, dist[i]);
    }
}

int main() {
    int visited[MAX_VERTICES] = {0};

    // Perform DFS traversal starting from vertex 0
    printf("DFS Traversal: ");
    DFS(graph, visited, 0);
    printf("\n");

    // Reset visited array for BFS
    for (int i = 0; i < MAX_VERTICES; i++) {
        visited[i] = 0;
    }

    // Perform BFS traversal starting from vertex 0
    printf("BFS Traversal: ");
    BFS(graph, visited, 0);
    printf("\n");

    // Perform Dijkstra's Algorithm starting from vertex 0
    Dijkstra(graph, 0);

    return 0;
```

```
}
```

Output :

```
DFS Traversal: 0 1 2 5 4 3

BFS Traversal: 0 1 2 4 3 5

Dijkstra's Algorithm (Shortest Path from Vertex 0):
Shortest distances from vertex 0:
To 0: 0
To 1: 7
To 2: 9
To 3: 20
To 4: 20
To 5: 11
```