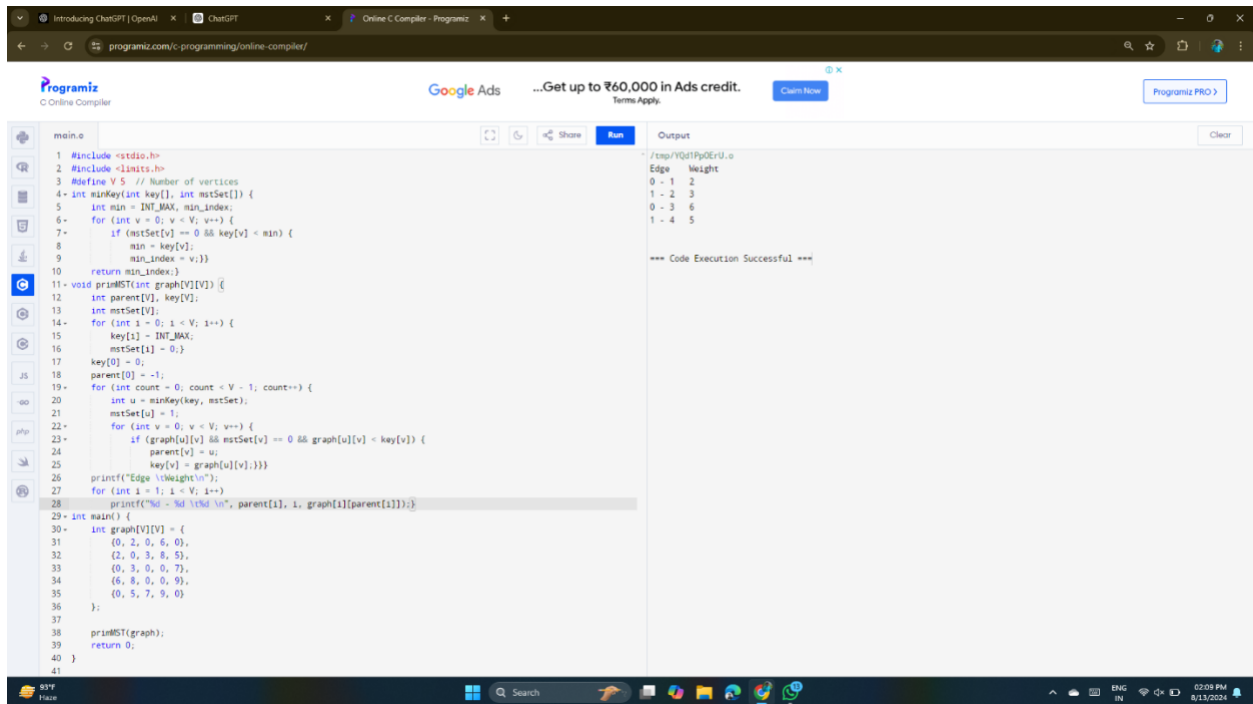


1. C Code for MINIMUM SPANNING TREE



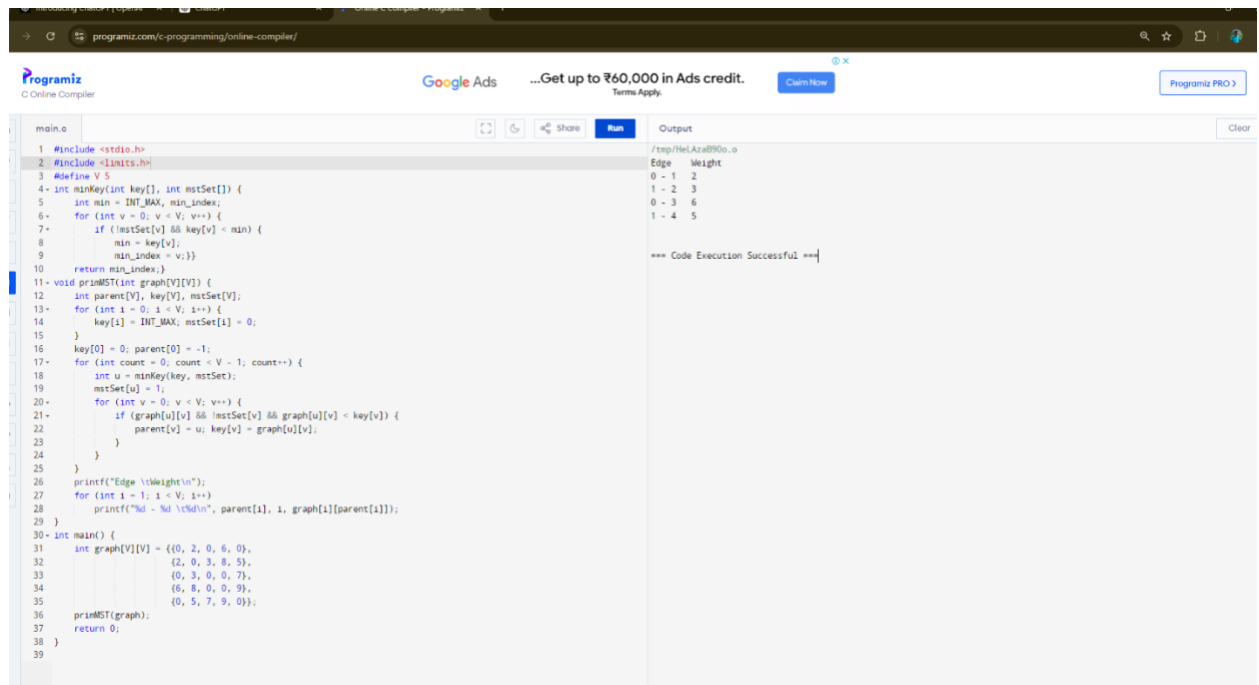
The screenshot shows a web browser with the URL `programiz.com/c-programming/online-compiler/`. The page features a Google Ads banner at the top. The main content area displays a C program for finding a Minimum Spanning Tree (MST) using Kruskal's algorithm. The code is in a file named `main.c` and includes standard headers, defines the number of vertices `V` as 5, and uses arrays for `key`, `minSet`, `parent`, and `graph`. The `minKey` function returns the index of the minimum key value in the `minSet` array. The `primMST` function implements Kruskal's algorithm by sorting edges by weight and adding them to the MST if they do not create a cycle. The `main` function initializes the graph and calls `primMST`. The output shows the edges of the MST and their weights.

```
1 #include <stdio.h>
2 #include <limits.h>
3 #define V 5 // Number of vertices
4 int minKey(int key[], int minSet[]) {
5     int min = INT_MAX, min_index;
6     for (int v = 0; v < V; v++) {
7         if (minSet[v] == 0 && key[v] < min) {
8             min = key[v];
9             min_index = v;
10        }
11    }
12    return min_index;
13}
14 void primMST(int graph[V][V]) {
15    int parent[V], key[V], minSet[V];
16    for (int i = 0; i < V; i++) {
17        key[i] = INT_MAX;
18        minSet[i] = 0;
19    }
20    key[0] = 0;
21    parent[0] = -1;
22    for (int count = 0; count < V - 1; count++) {
23        int u = minKey(key, minSet);
24        minSet[u] = 1;
25        for (int v = 0; v < V; v++) {
26            if (graph[u][v] && minSet[v] == 0 && graph[u][v] < key[v]) {
27                parent[v] = u;
28                key[v] = graph[u][v];
29            }
30        }
31        printf("Edge \tWeight\n");
32        for (int i = 1; i < V; i++)
33            printf("%d - %d \t%d\n", parent[i], i, graph[i][parent[i]]);
34    }
35}
36 int main() {
37    int graph[V][V] = {
38        {0, 2, 0, 6, 0},
39        {2, 0, 3, 8, 5},
40        {0, 3, 0, 0, 7},
41        {6, 8, 0, 0, 9},
42        {0, 5, 7, 9, 0}
43    };
44    primMST(graph);
45    return 0;
46}
```

Output:

```
Edge Weight
0 - 1 2
1 - 2 3
0 - 3 6
1 - 4 5
*** Code Execution Successful ***
```

2. C Code for PRIM'S ALGORITHM



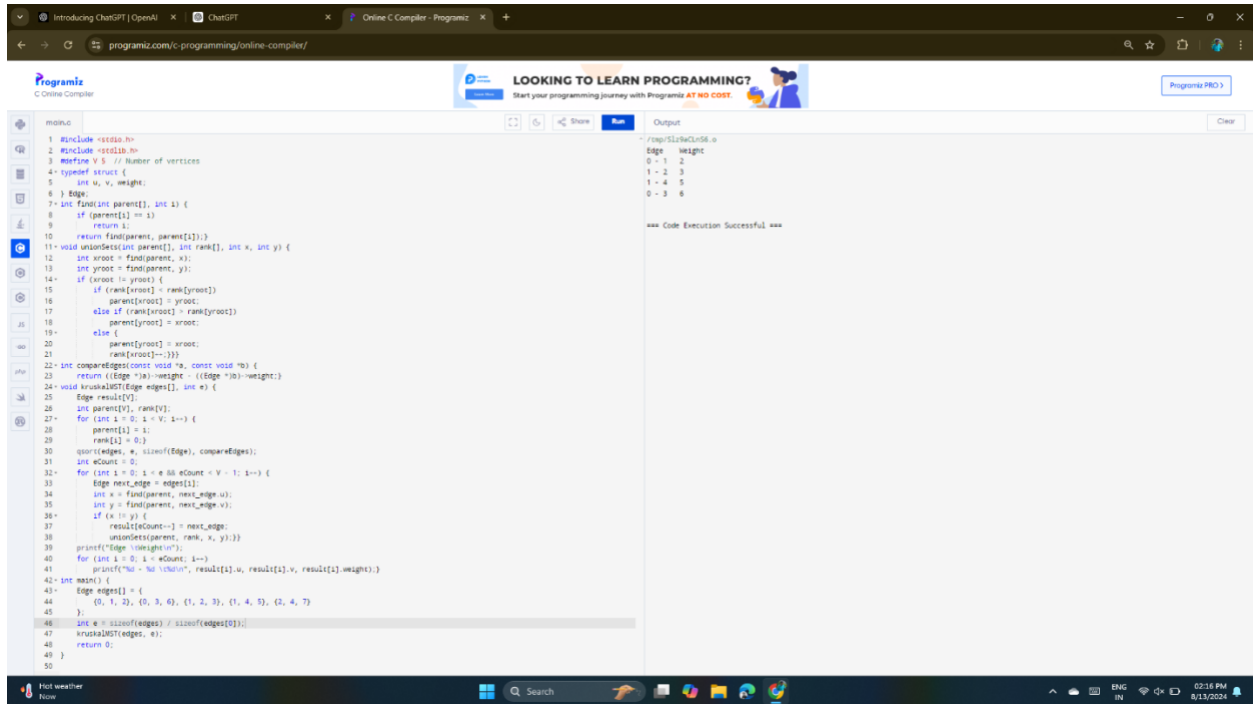
The screenshot shows the same online C compiler interface as above, but with a different C program for finding a Minimum Spanning Tree (MST) using Prim's algorithm. The code is in a file named `main.c` and includes standard headers, defines the number of vertices `V` as 5, and uses arrays for `key`, `minSet`, `parent`, and `graph`. The `minKey` function returns the index of the minimum key value in the `minSet` array. The `primMST` function implements Prim's algorithm by starting from a root vertex (0) and adding the minimum weight edge that connects a vertex in the MST to a vertex not in the MST. The `main` function initializes the graph and calls `primMST`. The output shows the edges of the MST and their weights.

```
1 #include <stdio.h>
2 #include <limits.h>
3 #define V 5
4 int minKey(int key[], int minSet[]) {
5     int min = INT_MAX, min_index;
6     for (int v = 0; v < V; v++) {
7         if (!minSet[v] && key[v] < min) {
8             min = key[v];
9             min_index = v;
10        }
11    }
12    return min_index;
13}
14 void primMST(int graph[V][V]) {
15    int parent[V], key[V], minSet[V];
16    for (int i = 0; i < V; i++) {
17        key[i] = INT_MAX;
18        minSet[i] = 0;
19    }
20    key[0] = 0;
21    parent[0] = -1;
22    for (int count = 0; count < V - 1; count++) {
23        int u = minKey(key, minSet);
24        minSet[u] = 1;
25        for (int v = 0; v < V; v++) {
26            if (!minSet[v] && graph[u][v] < key[v]) {
27                parent[v] = u;
28                key[v] = graph[u][v];
29            }
30        }
31    }
32    printf("Edge \tWeight\n");
33    for (int i = 1; i < V; i++)
34        printf("%d - %d \t%d\n", parent[i], i, graph[i][parent[i]]);
35}
36 int main() {
37    int graph[V][V] = {
38        {0, 2, 0, 6, 0},
39        {2, 0, 3, 8, 5},
40        {0, 3, 0, 0, 7},
41        {6, 8, 0, 0, 9},
42        {0, 5, 7, 9, 0}
43    };
44    primMST(graph);
45    return 0;
46}
```

Output:

```
Edge Weight
0 - 1 2
1 - 2 3
0 - 3 6
1 - 4 5
*** Code Execution Successful ***
```

3. C Code for KRUSHKAL ALGORITHM



The screenshot displays an online C compiler interface with the following components:

- Browser Tabs:** Includes tabs for "Introducing ChatGPT", "ChatGPT", and "Online C Compiler - Programiz".
- Programiz Header:** Features the "Programiz C Online Compiler" logo, a "LOOKING TO LEARN PROGRAMMING?" banner with a "Programiz PRO" button, and a "Start your programming journey with Programiz AT NO COST" message.
- Code Editor:** Contains the C code for Kruskal's Algorithm. The code defines a graph structure, implements a 'find' function for disjoint sets, a 'union' function to merge sets, and a 'kruskalMST' function that sorts edges and constructs the minimum spanning tree. The graph has 5 vertices and 7 edges with weights: (0,1)=2, (0,3)=6, (1,2)=3, (1,4)=5, (2,4)=7, (2,3)=4, and (3,4)=7.
- Output Panel:** Shows the execution results, including the file path "/tmp/1128c0r0d.o", the output of the 'find' function for each vertex, and the final MST edges and their weights: "Edge weight", "0 - 1 2", "1 - 2 3", "1 - 4 5", and "0 - 3 6". A status message "*** Code Execution Successful ***" is also present.
- System Bar:** The bottom of the image shows a Windows taskbar with a search bar, application icons, and a system clock indicating 02:16 PM on 8/13/2024.