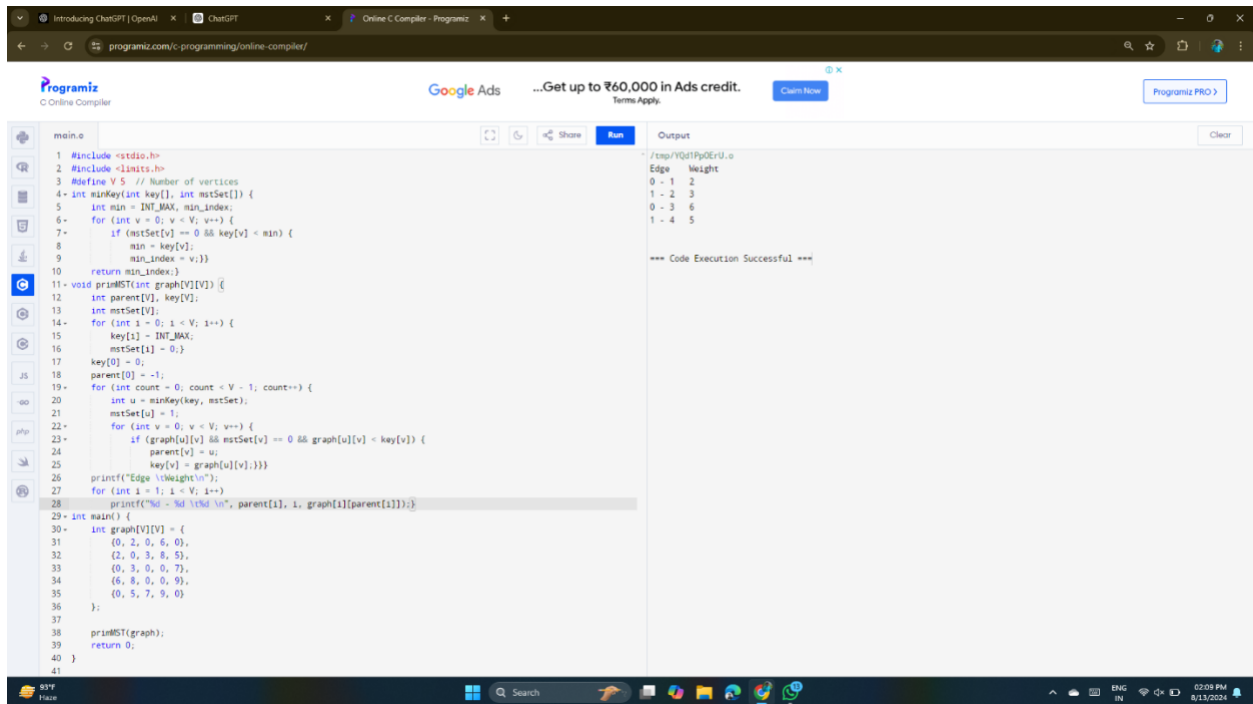


1. C Code for MINIMUM SPANNING TREE



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 element in `minSet`. 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.

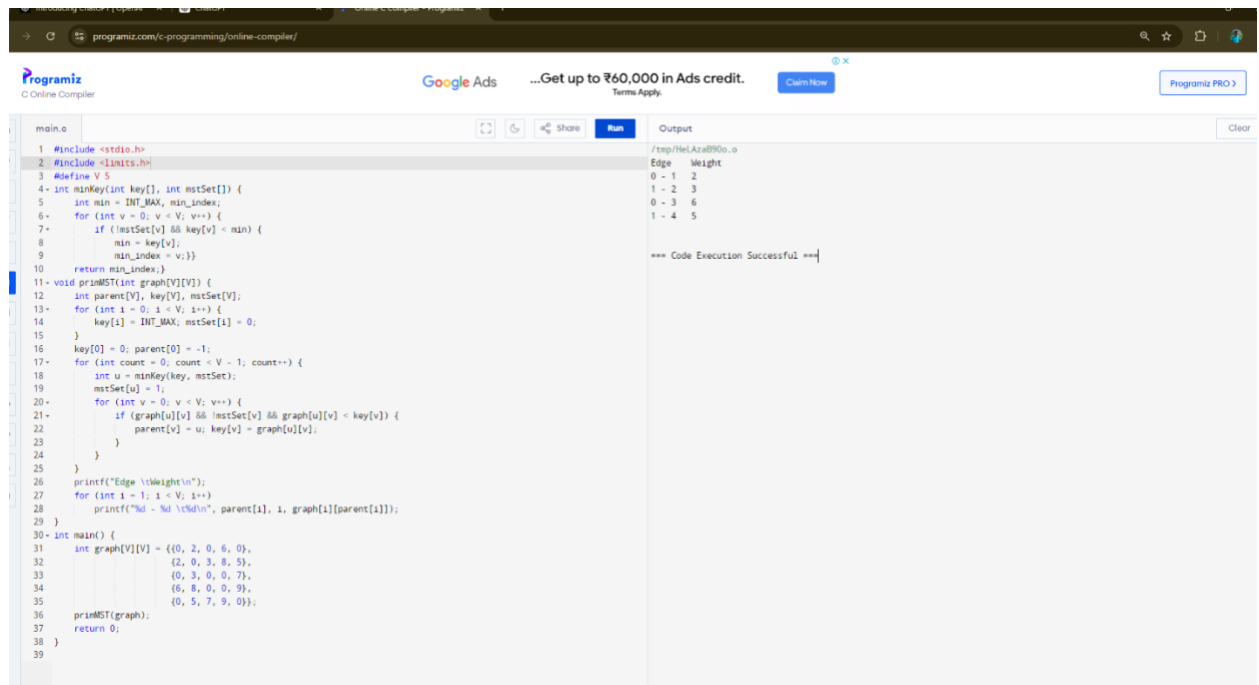
```
1 #include <stdio.h>
2 #include <limits.h>
3 #define V 5 // Number of vertices
4 int minKey(int key[], int minSet[]) {
5     int min = INT_MAX, min_index;
6     for (int v = 0; v < V; v++) {
7         if (minSet[v] == 0 && key[v] < min) {
8             min = key[v];
9             min_index = v;
10        }
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 element in `minSet`. The `primMST` function implements Prim's algorithm by starting from a single vertex 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.

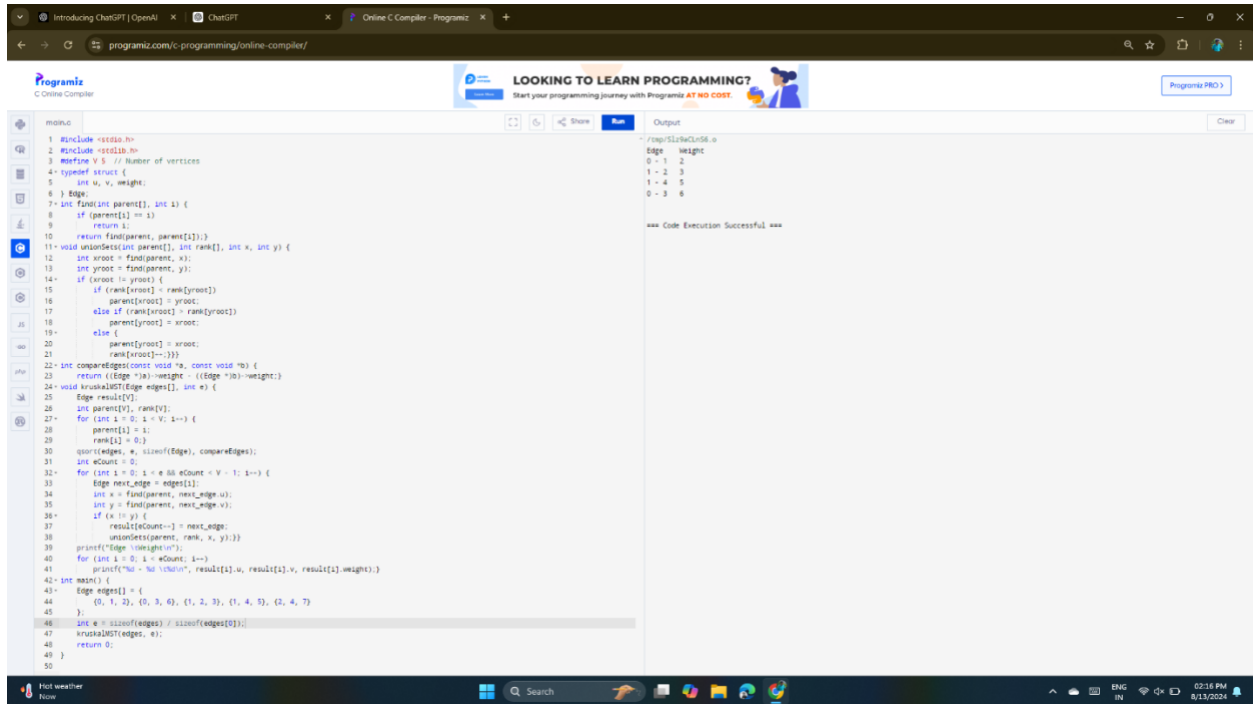
```
1 #include <stdio.h>
2 #include <limits.h>
3 #define V 5
4 int minKey(int key[], int minSet[]) {
5     int min = INT_MAX, min_index;
6     for (int v = 0; v < V; v++) {
7         if (!minSet[v] && key[v] < min) {
8             min = key[v];
9             min_index = v;
10        }
11    }
12    return 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 a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features a header for "Programiz C Online Compiler" and a promotional banner for "LOOKING TO LEARN PROGRAMMING?". The main content area is divided into two panels: a code editor on the left and an output console on the right.

The code editor contains the following C code for Kruskal's Algorithm:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define V 5 // Number of vertices
4 typedef struct {
5     int u, v, weight;
6 } Edge;
7 int find(int parent[], int i) {
8     if (parent[i] == i)
9         return i;
10    return find(parent, parent[i]);
11 void unionSets(int parent[], int rank[], int x, int y) {
12     int xroot = find(parent, x);
13     int yroot = find(parent, y);
14     if (xroot == yroot) {
15         return;
16     }
17     if (rank[xroot] < rank[yroot])
18         parent[xroot] = yroot;
19     else if (rank[xroot] > rank[yroot])
20         parent[yroot] = xroot;
21     else {
22         parent[xroot] = xroot;
23         rank[xroot]++;
24     }
25 int compareEdges(const void *a, const void *b) {
26     return ((Edge *)a)->weight - ((Edge *)b)->weight;
27 void kruskalMST(Edge edges[], int e) {
28     Edge result[V];
29     int parent[V], rank[V];
30     for (int i = 0; i < V; i++) {
31         parent[i] = i;
32         rank[i] = 0;
33     }
34     qsort(edges, e, sizeof(Edge), compareEdges);
35     int eCount = 0;
36     for (int i = 0; i < e && eCount < V - 1; i++) {
37         Edge nextEdge = edges[i];
38         int u = find(parent, nextEdge.u);
39         int v = find(parent, nextEdge.v);
40         if (u != v) {
41             result[eCount++] = nextEdge;
42             unionSets(parent, rank, u, v);
43             printf("Edge %d\n", i);
44         }
45     }
46     for (int i = 0; i < eCount; i++) {
47         printf("%d - %d\n", result[i].u, result[i].v, result[i].weight);
48     }
49 int main() {
50     Edge edges[] = {
51         {0, 1, 2}, {0, 3, 4}, {1, 2, 3}, {1, 4, 5}, {2, 4, 7}
52     };
53     int e = sizeof(edges) / sizeof(edges[0]);
54     kruskalMST(edges, e);
55     return 0;
56 }
```

The output console shows the following text:

```
./src/11srcUnit6.o
Edge weight
0 - 1 2
1 - 2 3
1 - 4 5
0 - 3 4
*** Code Execution Successful ***
```

The Windows taskbar at the bottom indicates the system time as 02:16 PM on 8/13/2024.