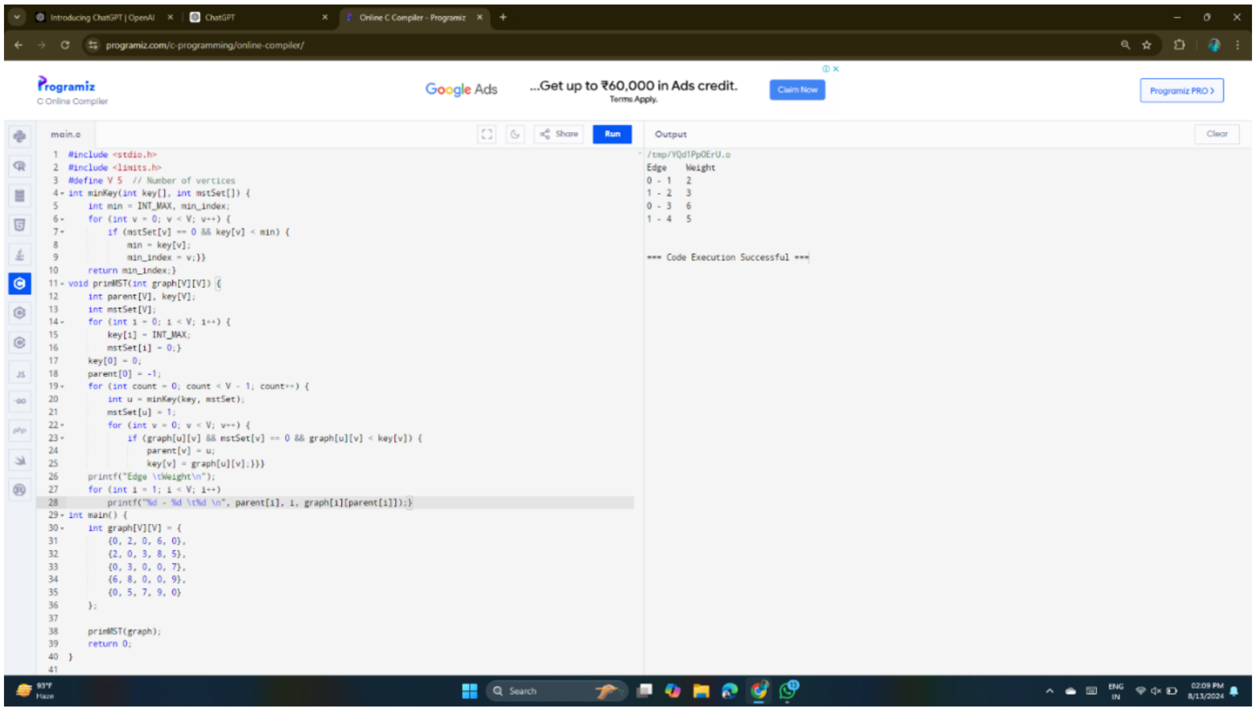
1. C Code for MINIMUM SPANNING TREE



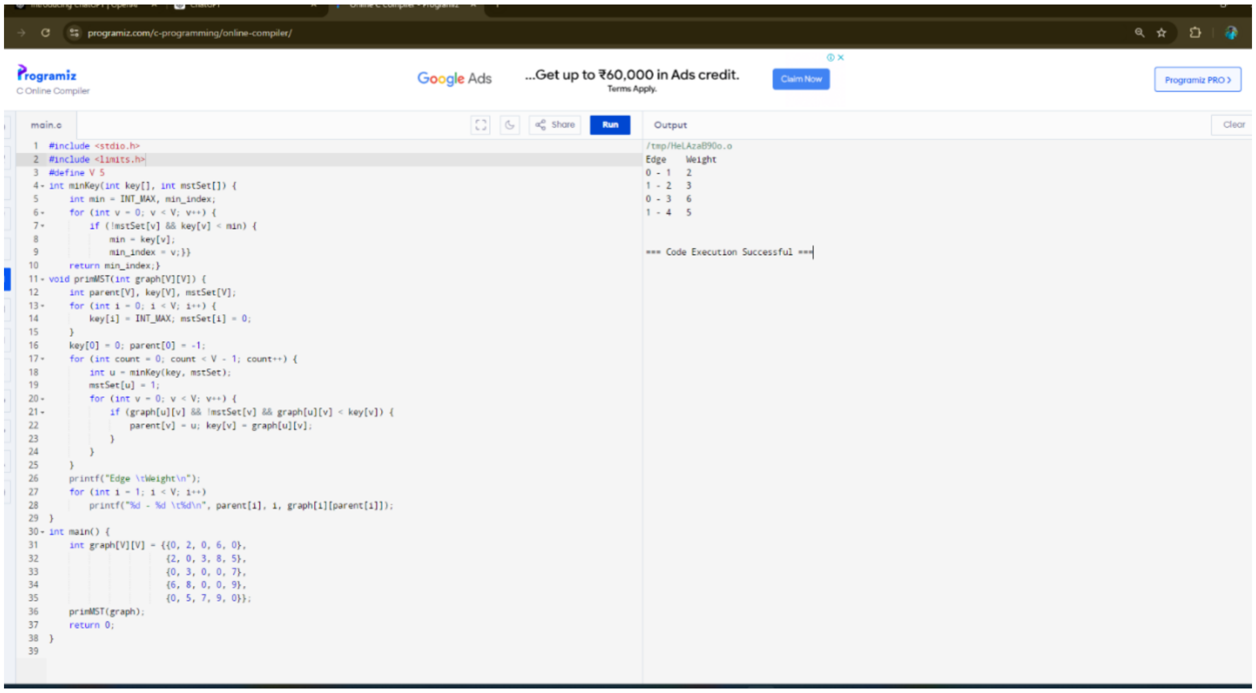
```c
#include <stdio.h>
#include <limits.h>
#define V 5  // Number of vertices
int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++) {
        if (mstSet[v] == 0 && key[v] < min) {
            min = key[v];
            min_index = v;}}
    return min_index;}
void primMST(int graph[V][V]) {
    int parent[V], key[V];
    int mstSet[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;}
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][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]]);}
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;
}
```

```
/tmp/YQd1Pp0ErU.o
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5


=== Code Execution Successful ===
```

2. C Code for PRIMS ALGORITHM

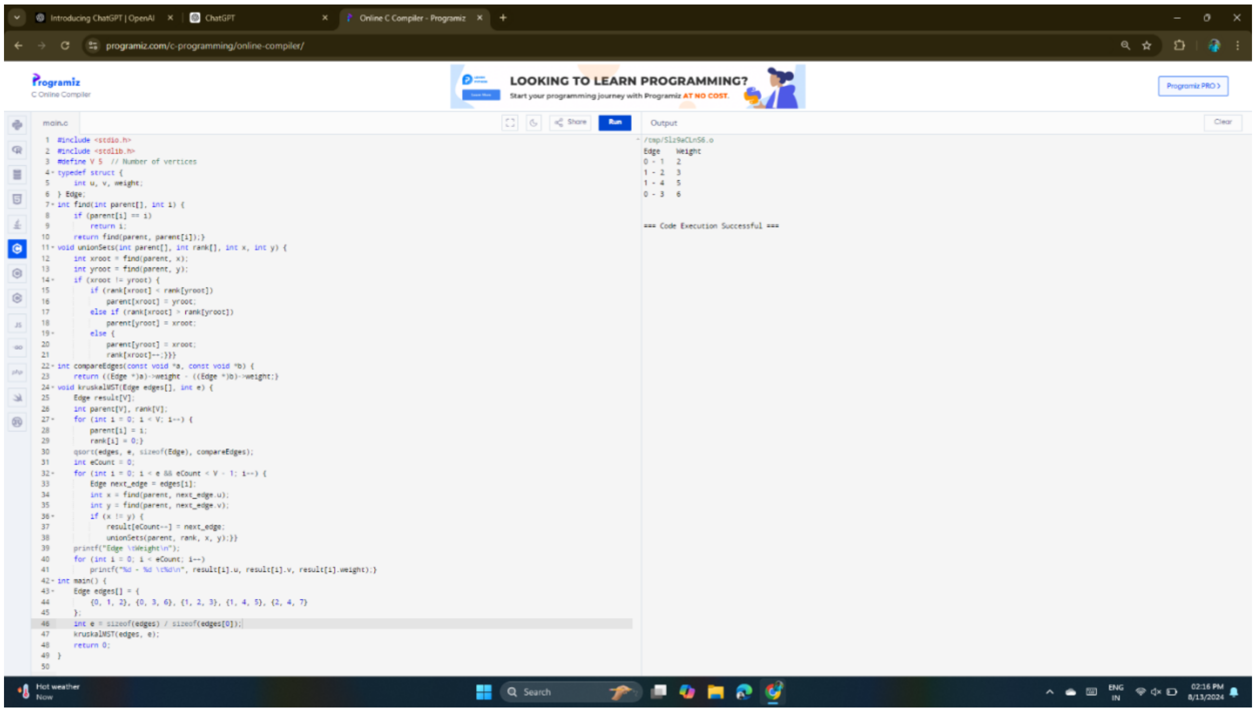


```c
#include <stdio.h>
#include <limits.h>
#define V 5
int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++) {
        if (!mstSet[v] && key[v] < min) {
            min = key[v];
            min_index = v;}}
    return min_index;}
void primMST(int graph[V][V]) {
    int parent[V], key[V], mstSet[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX; mstSet[i] = 0;
    }
    key[0] = 0; parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u; key[v] = graph[u][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]]);
}
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;
}
```

```
/tmp/HeLAzaB90o.o
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5


=== Code Execution Successful ===
```

3. C Code for KRUSHKAL ALGORITHM



```c
#include <stdio.h>
#include <stdlib.h>
#define V 5  // Number of vertices
typedef struct {
    int u, v, weight;
} Edge;
int find(int parent[], int i) {
    if (parent[i] == i)
        return i;
    return find(parent, parent[i]);}
void unionSets(int parent[], int rank[], int x, int y) {
    int xroot = find(parent, x);
    int yroot = find(parent, y);
    if (xroot != yroot) {
        if (rank[xroot] < rank[yroot])
            parent[xroot] = yroot;
        else if (rank[xroot] > rank[yroot])
            parent[yroot] = xroot;
        else {
            parent[yroot] = xroot;
            rank[xroot]++;}}}
int compareEdges(const void *a, const void *b) {
    return ((Edge *)a)->weight - ((Edge *)b)->weight;}
void kruskalMST(Edge edges[], int e) {
    Edge result[V];
    int parent[V], rank[V];
    for (int i = 0; i < V; i++) {
        parent[i] = i;
        rank[i] = 0;}
    qsort(edges, e, sizeof(Edge), compareEdges);
    int eCount = 0;
    for (int i = 0; i < e && eCount < V - 1; i++) {
        Edge next_edge = edges[i];
        int x = find(parent, next_edge.u);
        int y = find(parent, next_edge.v);
        if (x != y) {
            result[eCount++] = next_edge;
            unionSets(parent, rank, x, y);}}
    printf("Edge \tWeight\n");
    for (int i = 0; i < eCount; i++)
        printf("%d - %d \t%d\n", result[i].u, result[i].v, result[i].weight);}
int main() {
    Edge edges[] = {
        {0, 1, 2}, {0, 3, 6}, {1, 2, 3}, {1, 4, 5}, {2, 4, 7}
    };
    int e = sizeof(edges) / sizeof(edges[0]);
    kruskalMST(edges, e);
    return 0;
}
```

```
/tmp/Slz9aCLnS6.o
Edge    Weight
0 - 1   2
1 - 2   3
1 - 4   5
0 - 3   6


=== Code Execution Successful ===
```