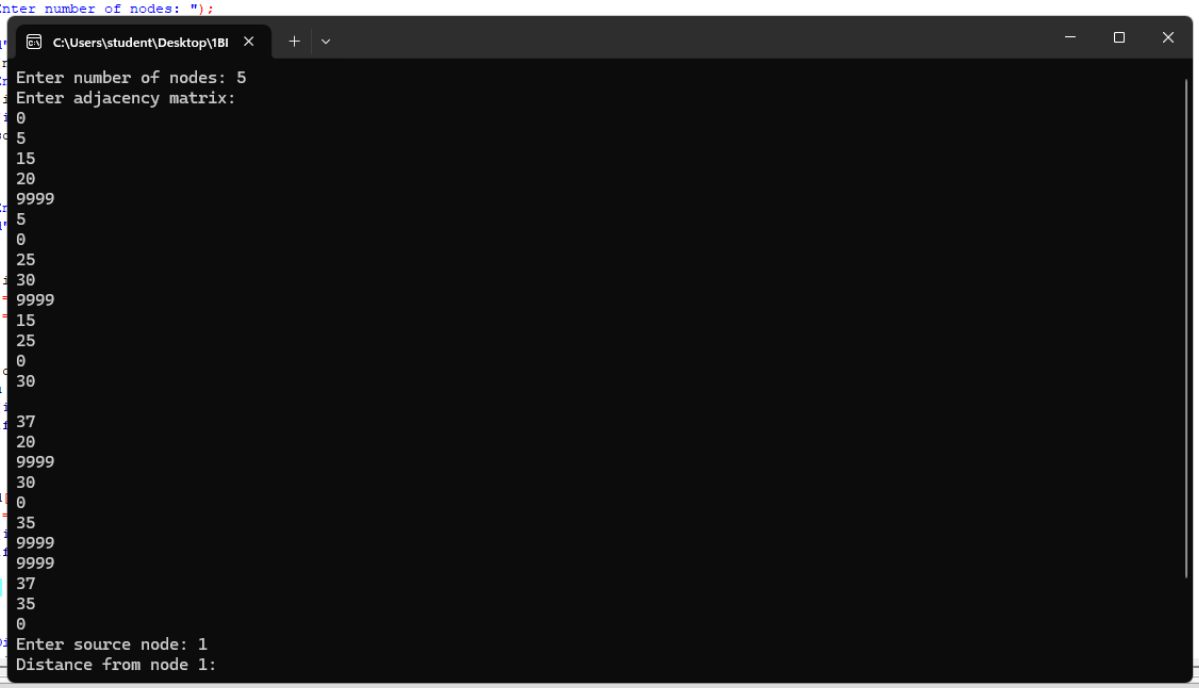
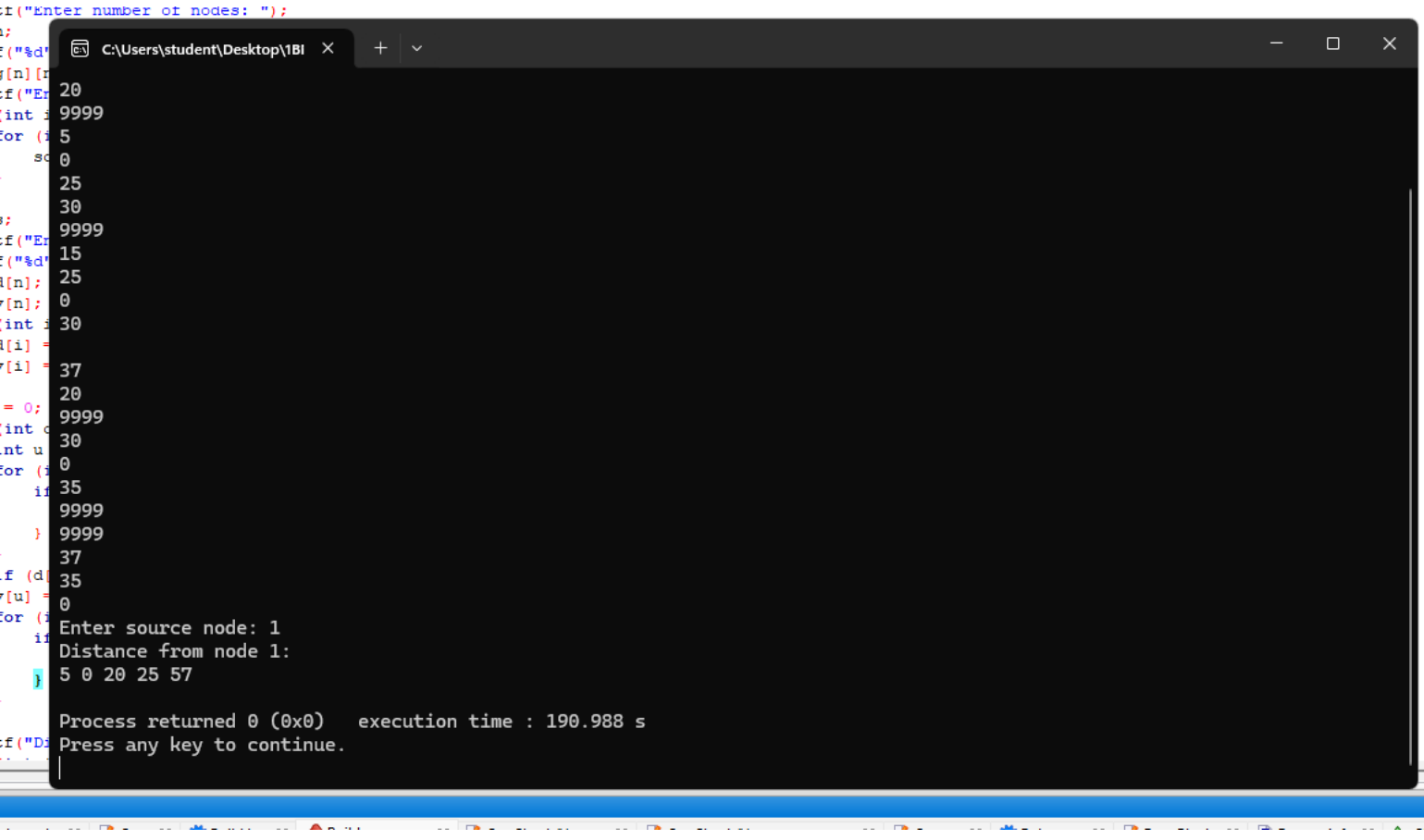
```c
#include <stdio.h>
#include <limits.h>

int main() {
    printf("Enter number of nodes: ");
    int n;
    scanf("%d", &n);
    int g[n][n];
    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &g[i][j]);
        }
    }
    int s;
    printf("Enter source node: ");
    scanf("%d", &s);
    int d[n];
    int v[n];
    for (int i = 0; i < n; i++) {
        d[i] = INT_MAX;
        v[i] = 0;
    }
    d[s] = 0;
    for (int count = 0; count < n - 1; count++) {
        int u = -1;
        for (int i = 0; i < n; i++) {
            if (!v[i] && (u == -1 || d[i] < d[u])) {
                u = i;
            }
        }
        if (d[u] == INT_MAX) break;
        v[u] = 1;
        for (int i = 0; i < n; i++) {
            if (g[u][i] && !v[i] && d[u] != INT_MAX && d[u] + g[u][i] < d[i]) {
                d[i] = d[u] + g[u][i];
            }
        }
    }
    printf("Distance from node %d:\n", s);
```

```
            scanf("%d", &g[i][j]);
        }
    }
    int s;
    printf("Enter source node: ");
    scanf("%d", &s);
    int d[n];
    int v[n];
    for (int i = 0; i < n; i++) {
        d[i] = INT_MAX;
        v[i] = 0;
    }
    d[s] = 0;
    for (int count = 0; count < n - 1; count++) {
        int u = -1;
        for (int i = 0; i < n; i++) {
            if (!v[i] && (u == -1 || d[i] < d[u])) {
                u = i;
            }
        }
        if (d[u] == INT_MAX) break;
        v[u] = 1;
        for (int i = 0; i < n; i++) {
            if (g[u][i] && !v[i] && d[u] != INT_MAX && d[u] + g[u][i] < d[i]) {
                d[i] = d[u] + g[u][i];
            }
        }
    }
    printf("Distance from node %d:\n", s);
    for (int i = 0; i < n; i++) {
        if (d[i] == INT_MAX) {
            printf("INF ");
        } else {
            printf("%d ", d[i]);
        }
    }
    printf("\n");
    return 0;
}
```

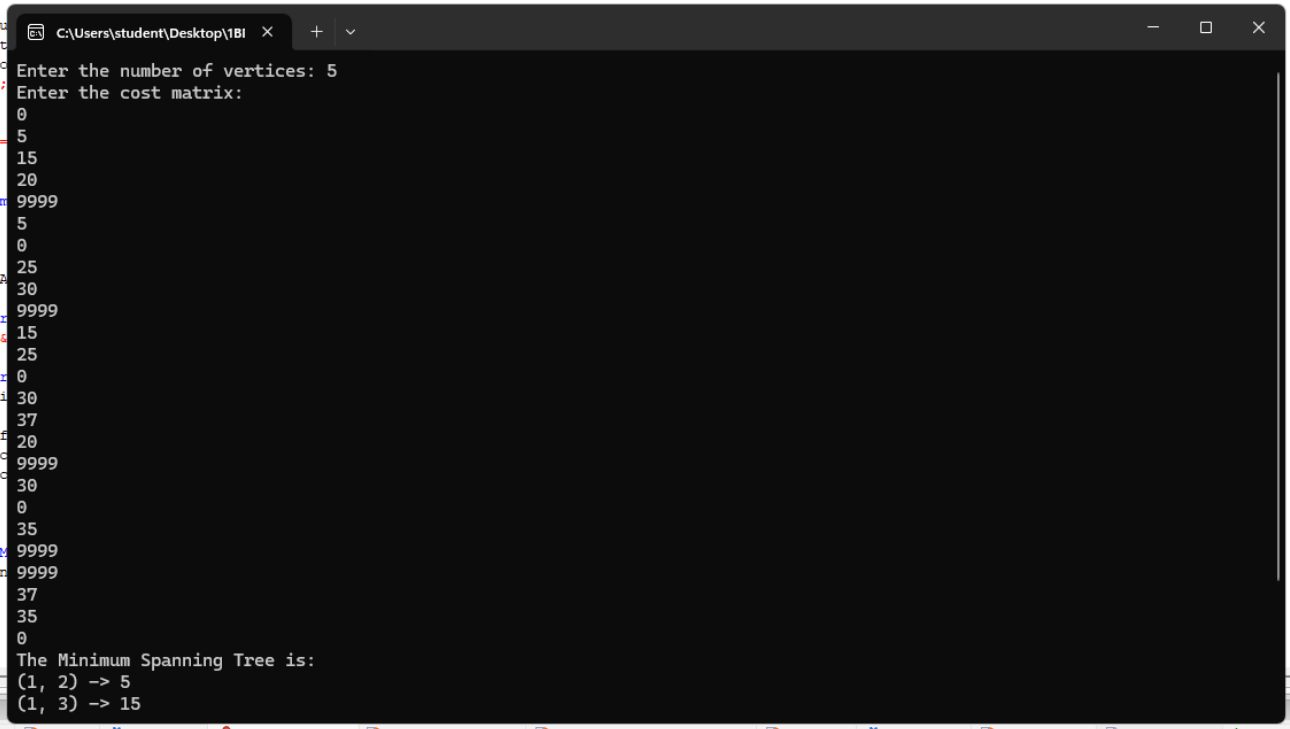

C:\Users\student\Desktop\1BI
Enter number of nodes: 5
Enter adjacency matrix:
0
5
15
20
9999
5
0
25
30
9999
15
25
0
30
37
20
9999
30
0
35
9999
9999
37
35
0
Enter source node: 1
Distance from node 1:

arch results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fortran info
Closed files list
Thread search


C:\Users\student\Desktop\1BI
20
9999
5
0
25
30
9999
15
25
0
30
37
20
9999
30
0
35
9999
9999
37
35
0
Enter source node: 1
Distance from node 1:
5 0 20 25 57
Process returned 0 (0x0) execution time : 190.988 s
Press any key to continue.

```c
#include <stdio.h>

#define MAX 30

typedef struct edge {
    int u, v, cost;
} Edge;

Edge edges[MAX];
int parent[MAX];

int find(int i) {
    while (parent[i])
        i = parent[i];
    return i;
}

int uni(int i, int j) {
    if (i != j) {
        parent[j] = i;
        return 1;
    }
    return 0;
}

void kruskals(int c[MAX][MAX], int n) {
    int i, j, u, v, a, b, min, ne = 0, mincost = 0;

    for (i = 1; i <= n; i++)
        parent[i] = 0;

    while (ne < n - 1) {
        min = 9999;
        for (i = 1; i <= n; i++) {
            for (j = 1; j <= n; j++) {
                if (c[i][j] < min) {
                    min = c[i][j];
                    u = a = i;
                    v = b = j;
                }
```

```
            }
        }
    }

    u = find(u);
    v = find(v);

    if (uni(u, v)) {
        printf("(%d, %d) -> %d\n", a, b, min);
        mincost += min;
        ne++;
    }

    c[a][b] = c[b][a] = 9999;
}

printf("Minimum Cost = %d\n", mincost);
}

int main() {
int c[MAX][MAX], n, i, j;

printf("Enter the number of vertices: ");
scanf("%d", &n);

printf("Enter the cost matrix:\n");
for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
        scanf("%d", &c[i][j]);
        if (c[i][j] == 0)
            c[i][j] = 9999;
    }
}

printf("The Minimum Spanning Tree is:\n");
kruskals(c, n);

return 0;
}
```



ults | Cccc | Build log | Build messages | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | DoxyBlocks | Fortran info | Closed files list | Thread search

essage
== Build file: "no target" in "no project" (compiler: unknown) ===
== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 4 second(s)) ===