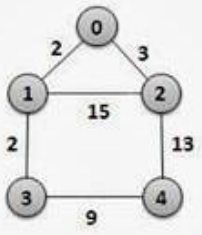
Programming Assignment 2 – CS3329

Student:

ID:

1. [10]Write a program to implement Dijkstra single source shorters path algorithm and show your solution on following graph (use the adjacency matrix representation as below). User will pick the source.



0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

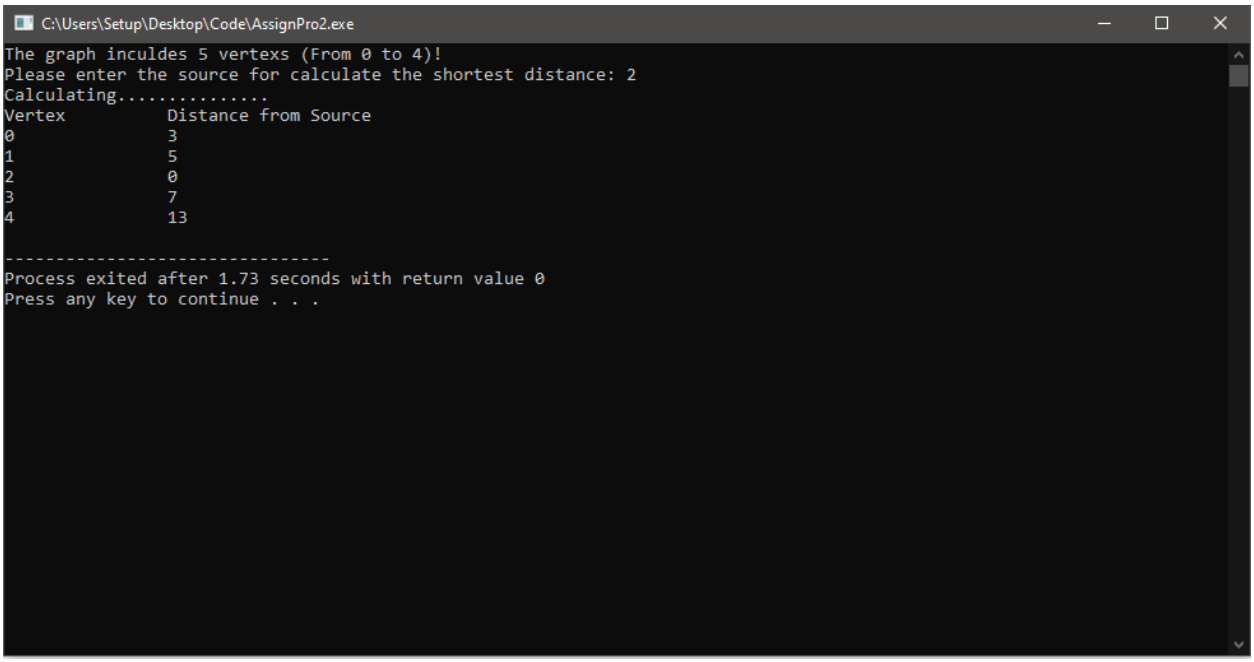
Adjacency Matrix Representation of

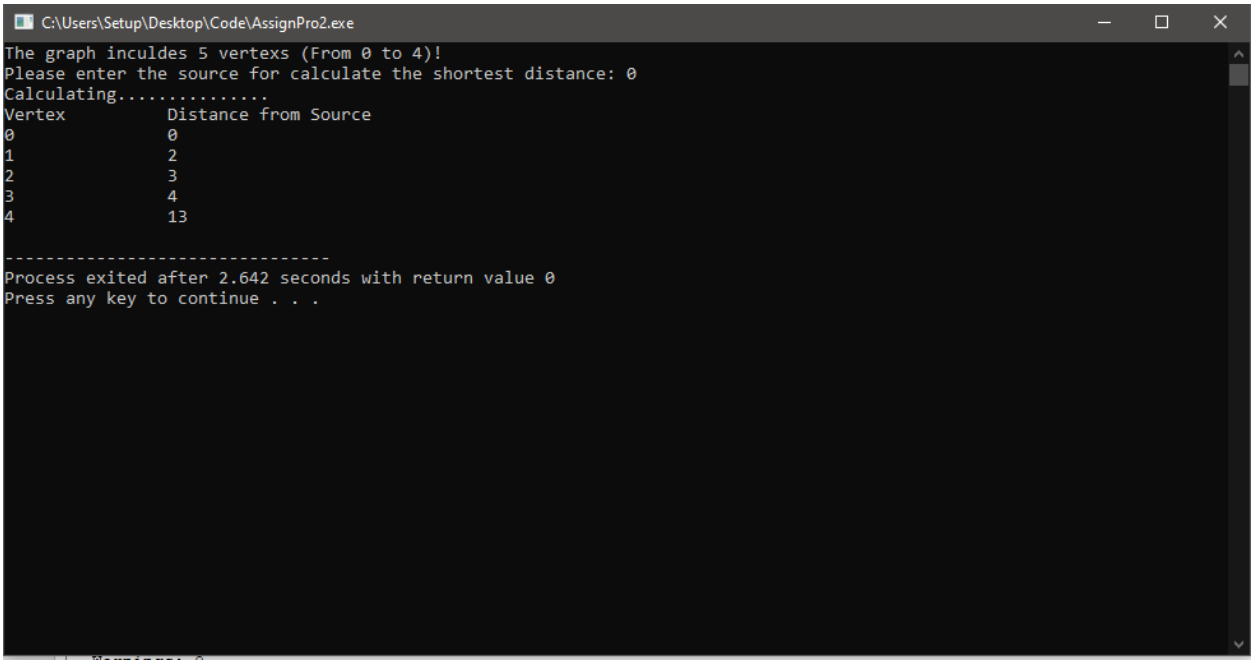
Code:

```
#include <iostream>
using namespace std;

#define Vertex 5

int minDistance(int dist[], bool sptSet[]) {
    int min = INT_MAX, min_index;
    for (int i=0; i<Vertex; i++)
        if (sptSet[i]==false && dist[i]<=min)
            min = dist[i], min_index = i;
    return min_index;
}

int displayResult(int dist[], int n) {
    cout << "Vertex\t\t" << left << "Distance from Source" << endl;;
    for (int i= 0; i<Vertex; i++)
        cout << i << "\t\t" << left << dist[i] << endl;
}


void dijkstra(int graph[Vertex][Vertex], int source) {
    int dist[Vertex];
    bool sptSet[Vertex];
    for (int i=0; i<Vertex; i++) {
        dist[i] = INT_MAX;
        sptSet[i] = false;
```

```
    }
    dist[source] = 0;
    for (int i=0; i<Vertex-1; i++) {
        int u = minDistance(dist, sptSet);
        sptSet[u] = true;
        for (int v=0; v<Vertex; v++)
            if (!sptSet[v] && graph[u][v] && dist[u]!=INT_MAX && dist[u]
+graph[u][v]<dist[v])
                dist[v] = dist[u] + graph[u][v];
    }
    displayResult(dist, Vertex);
}

int main() {
    int source;
    int graph[Vertex][Vertex] = {
        {0, 2, 3, 0, 0},
        {2, 0, 15, 2, 0},
        {3, 15, 0, 0, 13},
        {0, 2, 0, 0, 9},
        {0, 0, 13, 9, 0}
    };
    cout << "The graph inculdes 5 vertexs (From 0 to 4)!" << endl;
    cout << "Please enter the source for calculate the shortest distance:
";
    cin >> source;
    while (source<0 || source>4) {
        cout << "Invalid vertex!!! \nPlease choose one soure from 1 to 4:
";
        cin >> source;
    }
    cout << "Calculating.....\n";
    dijkstra(graph, source);
    return 0;
}
```

Output:


C:\Users\Setup\Desktop\Code\AssignPro2.exe
The graph inculdes 5 vertexs (From 0 to 4)!
Please enter the source for calculate the shortest distance: 2
Calculating.....
Vertex Distance from Source
0 3
1 5
2 0
3 7
4 13
Process exited after 1.73 seconds with return value 0
Press any key to continue . . .


C:\Users\Setup\Desktop\Code\AssignPro2.exe
The graph inculdes 5 vertexs (From 0 to 4)!
Please enter the source for calculate the shortest distance: 0
Calculating.....
Vertex Distance from Source
0 0
1 2
2 3
3 4
4 13
Process exited after 2.642 seconds with return value 0
Press any key to continue . . .

2. [10] Write a program to implement Knapsack problem in Solve. show your solution for following knapsack problem, show the objects picked and corresponding overall value of the sack:

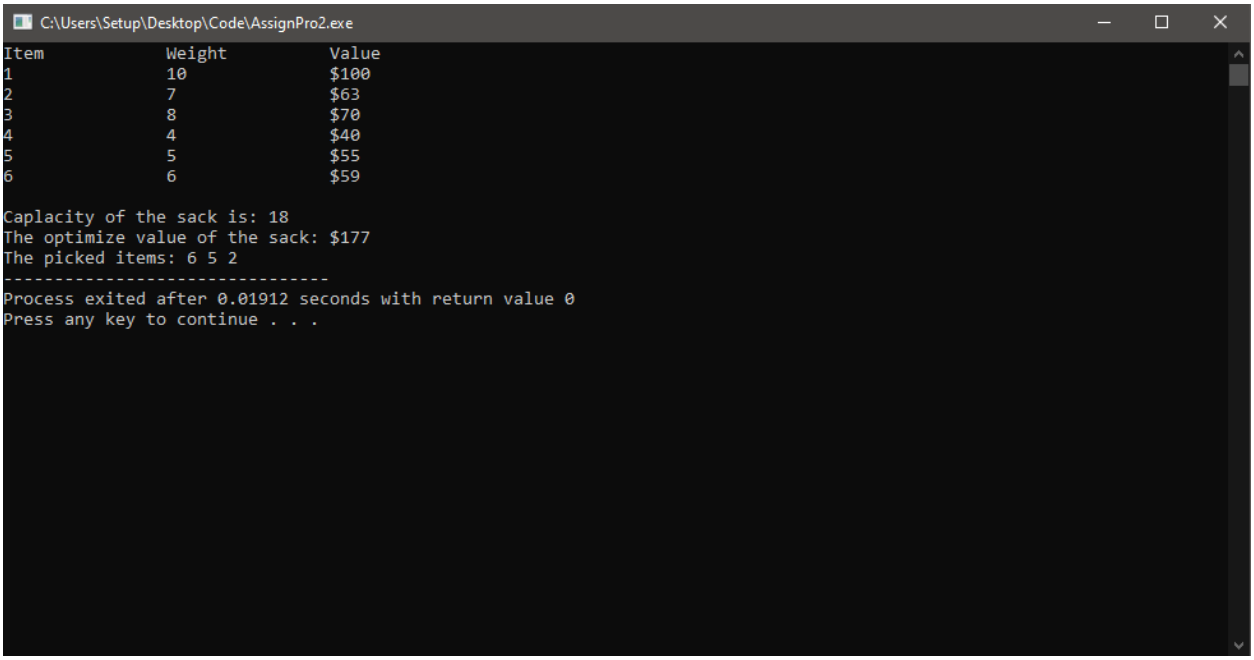
Item	Weight	Value	
1	10	$100	
2	7	$63	
3	8	$70	C = 18
4	4	$40	
5	5	$55	
6	6	$59	

Code:

```
#include <iostream>
using namespace std;

int max(int a, int b) {
    return (a > b) ? a : b;
}

void knapSack(int W, int wt[], int val[], int n) {
    int i, w;
    int K[n+1][W+1];
    for (i=0; i<=n; i++) {
        for (w=0; w<=W; w++) {
            if (i==0 ||w== 0)
                K[i][w] = 0;
            else if (wt[i-1] <= w)
                K[i][w] = max(val[i-1] + K[i-1][w -wt[i-1]], K[i-1][w]);
```

```
            else
                K[i][w] = K[i-1][w];
        }
    }

    int result = K[n][W];
    cout << "The optimize value of the sack: $" << result << endl;
    w = W;
    cout << "The picked items: ";
    for (i=n; i>0 && result>0; i--) {
        if (result == K[i-1][w])
            continue;
        else {
            cout << i << " ";
            result = result - val[i-1];
            w = w - wt[i-1];
        }
    }
}

int main() {
    int weight[] = {10, 7, 8, 4, 5, 6};
    int val[] = { 100, 63, 70, 40 ,55, 59 };
    int capacity = 18;
    int numOfItems = sizeof(val) / sizeof(val[0]);
    cout << left << "Item\t\t" << "Weight\t\t" << "Value" << endl;
    for (int i=0; i<numOfItems; i++) {
        cout << left << i+1 << "\t\t" << weight[i] <<"\t\t$" << val[i] <<
endl;
    }
    cout << "\nCaplacity of the sack is: " << capacity << endl;
    knapSack(capacity, weight, val, numOfItems);
    return 0;
}
```

Output:


C:\Users\Setup\Desktop\Code\AssignPro2.exe
Item Weight Value
1 10 $100
2 7 $63
3 8 $70
4 4 $40
5 5 $55
6 6 $59
Caplacity of the sack is: 18
The optimize value of the sack: $177
The picked items: 6 5 2
Process exited after 0.01912 seconds with return value 0
Press any key to continue . . .