**Question#1  
Task\_1\_a:**

#include <iostream>

using namespace std;

void dijkstraMatrix(int\*\* graph, int V, int source)

{

int\* dist = new int[V];

bool\* visited = new bool[V];

for (int i = 0; i < V; i++)

{

dist[i] = INT\_MAX;

visited[i] = false;

}

dist[source] = 0;

for (int count = 0; count < V - 1; count++)

{

int u = -1;

for (int i = 0; i < V; i++)

{

if (!visited[i])

{

if (u == -1 || dist[i] < dist[u])

{

u = i;

}

}

}

visited[u] = true;

for (int v = 0; v < V; v++)

{

if (!visited[v] && graph[u][v] && dist[u] != INT\_MAX)

{

if (dist[u] + graph[u][v] < dist[v])

{

dist[v] = dist[u] + graph[u][v];

}

}

}

}

for (int i = 0; i < V; i++)

{

cout << "Vertex " << i << " Distance: ";

if (dist[i] == INT\_MAX)

{

cout << "INF";

}

else

{

cout << dist[i];

}

cout << endl;

}

delete[] dist;

delete[] visited;

}

int main()

{

int V = 5;

int\*\* graph = new int\* [V];

for (int i = 0; i < V; i++)

{

graph[i] = new int[V];

for (int j = 0; j < V; j++)

{

graph[i][j] = 0;

}

}

graph[0][1] = 10;

graph[0][4] = 5;

graph[1][2] = 1;

graph[1][4] = 2;

graph[2][3] = 4;

graph[3][0] = 7;

graph[4][2] = 9;

graph[4][3] = 2;

dijkstraMatrix(graph, V, 0);

for (int i = 0; i < V; i++)

{

delete[] graph[i];

}

delete[] graph;

return 0;

}

**A screenshot of a computer

Description automatically generated  
Task\_1\_b:**

#include <iostream>

using namespace std;

struct Node

{

int vertex;

int weight;

Node\* next;

};

void dijkstraList(Node\*\* adjList, int V, int source)

{

int\* dist = new int[V];

bool\* visited = new bool[V];

for (int i = 0; i < V; i++)

{

dist[i] = INT\_MAX;

visited[i] = false;

}

dist[source] = 0;

for (int count = 0; count < V - 1; count++)

{

int u = -1;

for (int i = 0; i < V; i++)

{

if (!visited[i])

{

if (u == -1 || dist[i] < dist[u])

{

u = i;

}

}

}

visited[u] = true;

Node\* temp = adjList[u];

while (temp != nullptr)

{

int v = temp->vertex;

int weight = temp->weight;

if (!visited[v] && dist[u] != INT\_MAX)

{

if (dist[u] + weight < dist[v])

{

dist[v] = dist[u] + weight;

}

}

temp = temp->next;

}

}

for (int i = 0; i < V; i++)

{

cout << "Vertex " << i << " Distance: ";

if (dist[i] == INT\_MAX)

{

cout << "INF";

}

else

{

cout << dist[i];

}

cout << endl;

}

delete[] dist;

delete[] visited;

}

void addEdge(Node\*\* adjList, int u, int v, int weight)

{

Node\* newNode = new Node;

newNode->vertex = v;

newNode->weight = weight;

newNode->next = adjList[u];

adjList[u] = newNode;

newNode = new Node;

newNode->vertex = u;

newNode->weight = weight;

newNode->next = adjList[v];

adjList[v] = newNode;

}

int main()

{

int V = 5;

Node\*\* adjList = new Node \* [V];

for (int i = 0; i < V; i++)

{

adjList[i] = nullptr;

}

addEdge(adjList, 0, 1, 10);

addEdge(adjList, 0, 4, 5);

addEdge(adjList, 1, 2, 1);

addEdge(adjList, 1, 4, 2);

addEdge(adjList, 2, 3, 4);

addEdge(adjList, 3, 0, 7);

addEdge(adjList, 4, 2, 9);

addEdge(adjList, 4, 3, 2);

dijkstraList(adjList, V, 0);

for (int i = 0; i < V; i++)

{

Node\* temp = adjList[i];

while (temp != nullptr)

{

Node\* toDelete = temp;

temp = temp->next;

delete toDelete;

}

}

delete[] adjList;

return 0;

} **A screenshot of a computer

Description automatically generated**

**Question#2**

#include <iostream>

using namespace std;

struct Node

{

string key;

int value;

Node\* next;

};

class HashTable

{

private:

Node\* table[10];

int hashFunction(string key)

{

int sum = 0;

for (char c : key)

{

sum += c;

}

return sum % 10;

}

public:

HashTable()

{

for (int i = 0; i < 10; i++)

{

table[i] = NULL;

}

}

void insert(string key, int value)

{

int index = hashFunction(key);

Node\* newNode = new Node;

newNode->key = key;

newNode->value = value;

newNode->next = table[index];

table[index] = newNode;

}

void remove(string key)

{

int index = hashFunction(key);

Node\* temp = table[index], \* prev = NULL;

while (temp != NULL && temp->key != key)

{

prev = temp;

temp = temp->next;

}

if (temp == NULL)

{

return;

}

if (prev == NULL)

{

table[index] = temp->next;

}

else

{

prev->next = temp->next;

}

delete temp;

}

int search(string key)

{

int index = hashFunction(key);

Node\* temp = table[index];

while (temp != NULL)

{

if (temp->key == key)

{

return temp->value;

}

temp = temp->next;

}

return -1;

}

void display()

{

for (int i = 0; i < 10; i++)

{

Node\* temp = table[i];

cout << i << ": ";

while (temp != NULL)

{

cout << "(" << temp->key << ", " << temp->value << ") -> ";

temp = temp->next;

}

cout << "NULL" << endl;

}

}

};

int main()

{

HashTable ht;

ht.insert("apple", 10);

ht.insert("banana", 20);

ht.insert("cherry", 30);

ht.insert("date", 40);

ht.insert("apple", 50);

cout << "HashTable Contents:" << endl;

ht.display();

cout << "Search for 'banana': " << ht.search("banana") << endl;

cout << "Search for 'grape': " << ht.search("grape") << endl;

ht.remove("banana");

cout << "After removing 'banana':" << endl;

ht.display();

return 0;

}

**A screenshot of a computer

Description automatically generated**

**Question#3**

#include <iostream>

using namespace std;

class HashTable

{

private:

int keys[10];

string values[10];

int hashFunction(int key)

{

return key % 10;

}

public:

HashTable()

{

for (int i = 0; i < 10; i++)

{

keys[i] = -1;

values[i] = "";

}

}

void insert(int key, string value)

{

int index = hashFunction(key);

while (keys[index] != -1)

{

index = (index + 1) % 10;

}

keys[index] = key;

values[index] = value;

}

void remove(int key)

{

int index = hashFunction(key);

while (keys[index] != -1)

{

if (keys[index] == key)

{

keys[index] = -1;

values[index] = "";

return;

}

index = (index + 1) % 10;

}

}

string search(int key)

{

int index = hashFunction(key);

while (keys[index] != -1)

{

if (keys[index] == key)

{

return values[index];

}

index = (index + 1) % 10;

}

return "NOT FOUND";

}

void display()

{

for (int i = 0; i < 10; i++)

{

if (keys[i] != -1)

{

cout << keys[i] << ": " << values[i] << endl;

}

else

{

cout << i << ": NULL" << endl;

}

}

}

};

int main()

{

HashTable ht;

ht.insert(15, "apple");

ht.insert(25, "banana");

ht.insert(35, "cherry");

ht.insert(45, "date");

cout << "HashTable Contents:" << endl;

ht.display();

cout << "Search for key 25: " << ht.search(25) << endl;

cout << "Search for key 50: " << ht.search(50) << endl;

ht.remove(25);

cout << "After removing key 25:" << endl;

ht.display();

return 0;

}

**A screenshot of a computer

Description automatically generated**