Write a C Program to represent a Resource Allocation Graph using appropriate data

structures for nodes and edges.

Ans:

#include <stdio.h>

#include <stdlib.h>

// Structure to represent a node in the graph

typedef struct Node {

int id;

int availableResource;

struct Edge\* edge;

} Node;

// Structure to represent an edge in the graph

typedef struct Edge {

struct Node\* sourceNode;

struct Node\* destinationNode;

int weight;

struct Edge\* nextEdge;

} Edge;

// Function to create a new node with given id and available resource

Node\* createNode(int id, int availableResource) {

Node\* newNode = (Node\*) malloc(sizeof(Node));

newNode->id = id;

newNode->availableResource = availableResource;

newNode->edge = NULL;

return newNode;

}

// Function to create a new edge with given source node, destination node and weight

Edge\* createEdge(Node\* sourceNode, Node\* destinationNode, int weight) {

Edge\* newEdge = (Edge\*) malloc(sizeof(Edge));

newEdge->sourceNode = sourceNode;

newEdge->destinationNode = destinationNode;

newEdge->weight = weight;

newEdge->nextEdge = NULL;

return newEdge;

}

// Function to add an edge to the given node

void addEdge(Node\* node, Edge\* edge) {

if (node->edge == NULL) {

node->edge = edge;

} else {

Edge\* currEdge = node->edge;

while (currEdge->nextEdge != NULL) {

currEdge = currEdge->nextEdge;

}

currEdge->nextEdge = edge;

}

}

// Function to display the resource allocation graph

void displayRAG(Node\* nodes[], int numNodes) {

printf("Resource Allocation Graph:\n");

for (int i = 0; i < numNodes; i++) {

printf("Node %d (Available Resource: %d)\n", nodes[i]->id, nodes[i]->availableResource);

Edge\* currEdge = nodes[i]->edge;

while (currEdge != NULL) {

printf("Edge: %d -> %d (Weight: %d)\n", currEdge->sourceNode->id, currEdge->destinationNode->id, currEdge->weight);

currEdge = currEdge->nextEdge;

}

printf("\n");

}

}

int main() {

// Create nodes

Node\* node1 = createNode(1, 5);

Node\* node2 = createNode(2, 10);

Node\* node3 = createNode(3, 8);

Node\* node4 = createNode(4, 15);

// Create edges

Edge\* edge1 = createEdge(node1, node2, 3);

Edge\* edge2 = createEdge(node1, node3, 2);

Edge\* edge3 = createEdge(node2, node3, 4);

Edge\* edge4 = createEdge(node3, node4, 5);

// Add edges to respective nodes

addEdge(node1, edge1);

addEdge(node1, edge2);

addEdge(node2, edge3);

addEdge(node3, edge4);

// Create an array to store nodes

Node\* nodes[] = {node1, node2, node3, node4};

int numNodes = sizeof(nodes) / sizeof(nodes[0]);

// Display the resource allocation graph

displayRAG(nodes, numNodes);

return 0;

}