DSA FINAL ANSWERS

1. **Write a c++ program for DFS traversal for the below given graph.**

**Ans :-**

#include <iostream>

#include <vector>

using namespace std;

void DFS(int node, vector<vector<int>> &adj, vector<bool> &visited) {

visited[node] = true;

cout << node << " ";

for (int neighbor : adj[node]) {

if (!visited[neighbor]) {

DFS(neighbor, adj, visited);

}

}

}

int main() {

int vertices, edges;

cout << "Enter number of vertices: ";

cin >> vertices;

cout << "Enter number of edges: ";

cin >> edges;

vector<vector<int>> adj(vertices);

cout << "Enter edges (u v):" << endl;

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

int u, v;

cin >> u >> v;

adj[u].push\_back(v);

adj[v].push\_back(u);

}

vector<bool> visited(vertices, false);

cout << "DFS Traversal starting from node 0: ";

DFS(0, adj, visited);

return 0;

}

1. **Write a c++ program for BFS traversal for the below given graph.**

**Ans :-**

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void BFS(int start, vector<vector<int>> &adj, vector<bool> &visited) {

queue<int> q;

q.push(start);

visited[start] = true;

cout << "BFS Traversal: ";

while (!q.empty()) {

int node = q.front();

q.pop();

cout << node << " ";

for (int neighbor : adj[node]) {

if (!visited[neighbor]) {

q.push(neighbor);

visited[neighbor] = true;

}

} }

}

int main()

{

int vertices, edges;

cout << "Enter number of vertices: ";

cin >> vertices;

cout << "Enter number of edges: ";

cin >> edges;

vector<vector<int>> adj(vertices);

cout << "Enter edges (u v):" << endl;

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

int u, v;

cin >> u >> v;

adj[u].push\_back(v);

adj[v].push\_back(u); }

vector<bool> visited(vertices, false);

cout << "Enter the starting node for BFS: ";

int startNode;

cin >> startNode;

BFS(startNode, adj, visited);

return 0;

}

1. **Write a c++ program for a singly linked list where the first node will contain your PRN, second node will contain your name and the third node will contain your age.**

**Ans :-**

#include <iostream>

#include <string>

using namespace std;

struct Node {

int prn;

string name;

int age;

Node\* next;

};

Node\* createNode(int prn, string name, int age) {

Node\* newNode = new Node();

newNode->prn = prn;

newNode->name = name;

newNode->age = age;

newNode->next = nullptr;

return newNode;

}

void displayList(Node\* head) {

Node\* temp = head;

cout << "Singly Linked List:" << endl;

while (temp != nullptr) {

cout << "PRN: " << temp->prn << ", Name: " << temp->name << ", Age: " << temp->age

<< endl;

temp = temp->next;

}

}

int main() {

Node\* head = createNode(2046, "Aditya Shinde", 20);

head->next = createNode(2053, "Yash Shinde", 19);

head->next->next = createNode(2054, "Rohan Patil", 18);

displayList(head);

return 0;

}

1. **Write a c++ program using the vector/array data structure where you will push 5 numbers.The numbers would be your RollNumber, RollNumber-1, RollNumber-2, RollNumber-3, RollNumber-4. Print the size of the array/vector.**

**Ans :-**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> rollNumbers;

int rollNumber;

cout << "Enter your Roll Number: ";

cin >> rollNumber;

for (int i = 0; i < 5; i++) {

rollNumbers.push\_back(rollNumber - i);

}

cout << "Size of the vector: " << rollNumbers.size() << endl;

cout << "Roll Numbers in the vector: ";

for (int num : rollNumbers) {

cout << num << " ";

}

cout << endl;

return 0;

}

1. **Write a c++ program for construction of a Binary Tree. Print out the Binary Tree.**

**6. Write a c++ program for construction of a Binary Tree.**

**Ans :-**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* left;

Node\* right;

Node(int value) {

data = value;

left = nullptr;

right = nullptr;

}

};

void printInOrder(Node\* root) {

if (root == nullptr) {

return;

}

printInOrder(root->left);

cout << root->data << " ";

printInOrder(root->right);

}

int main() {

Node\* root = new Node(1);

root->left = new Node(2);

root->right = new Node(3);

root->left->left = new Node(4);

root->left->right = new Node(5);

root->right->left = new Node(6);

root->right->right = new Node(7);

cout << "Binary Tree in Inorder Traversal: ";

printInOrder(root);

cout << endl;

return 0;

}

1. **Write a c++ program for construction of a Binary Search Tree.**

**Ans :-**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* left;

Node\* right;

Node(int value) {

data = value;

left = nullptr;

right = nullptr;

}

};

Node\* insert(Node\* root, int value) {

if (root == nullptr) {

return new Node(value); }

if (value < root->data) {

root->left = insert(root->left, value);

} else {

root->right = insert(root->right, value); }

return root; }

void printInOrder(Node\* root) {

if (root == nullptr) {

return; }

printInOrder(root->left);

cout << root->data << " ";

printInOrder(root->right);

}

int main() {

Node\* root = nullptr;

root = insert(root, 50);

root = insert(root, 30);

root = insert(root, 70);

root = insert(root, 20);

root = insert(root, 40);

root = insert(root, 60);

root = insert(root, 80);

cout << "Binary Search Tree in Inorder Traversal: ";

printInOrder(root);

cout << endl;

return 0;

}

**8. Write a c++ program to print the inorder sequence of a Binary search tree.**

**9. Write a c++ program to print the preorder sequence of a Binary search tree.**

**10. Write a c++ program to print the postorder sequence of a Binary search tree.**

**Ans :-**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* left;

Node\* right;

Node(int value) {

data = value;

left = nullptr;

right = nullptr;

}

};

Node\* insert(Node\* root, int value) {

if (root == nullptr) {

return new Node(value);

}

if (value < root->data) {

root->left = insert(root->left, value);

} else {

root->right = insert(root->right, value);

}

return root;

}

void printInOrder(Node\* root) {

if (root == nullptr) {

return;

}

printInOrder(root->left);

cout << root->data << " ";

printInOrder(root->right);

}

void printPreOrder(Node\* root) {

if (root == nullptr) {

return;

}

cout << root->data << " ";

printPreOrder(root->left);

printPreOrder(root->right);

}

void printPostOrder(Node\* root) {

if (root == nullptr) {

return;

}

printPostOrder(root->left);

printPostOrder(root->right);

cout << root->data << " ";

}

int main() {

Node\* root = nullptr;

root = insert(root, 50);

root = insert(root, 30);

root = insert(root, 70);

root = insert(root, 20);

root = insert(root, 40);

root = insert(root, 60);

root = insert(root, 80);

cout << "Inorder Traversal: ";

printInOrder(root);

cout << endl;

cout << "Preorder Traversal: ";

printPreOrder(root);

cout << endl;

cout << "Postorder Traversal: ";

printPostOrder(root);

cout << endl;

return 0;

}

**11. Write a c++ program to store a graph using adjacency list and print the adjacency list.**

**Ans :-**

#include <iostream>

#include <vector>

class DirectedGraph {

private:

int numVertices;

std::vector<std::vector<int>> adjacencyList;

public:

DirectedGraph(int vertices) {

numVertices = vertices;

adjacencyList.resize(vertices);

}

void addEdge(int src, int dest) {

adjacencyList[src].push\_back(dest);

}

void printGraph() {

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

std::cout << "Vertex " << i << " has outgoing edges to: ";

for (int neighbor : adjacencyList[i]) {

std::cout << neighbor << " ";

}

std::cout << "\n";

}

}

};

int main() {

DirectedGraph graph(5);

graph.addEdge(0, 1);

graph.addEdge(0, 3);

graph.addEdge(1, 2);

graph.addEdge(2, 3);

graph.addEdge(2, 4);

graph.addEdge(3, 4);

graph.printGraph();

return 0;

}

**12. Write a c++ program to store a graph using an adjacency matrix and print the adjacency matrix.**

**Ans :-**

#include <iostream>

using namespace std;

const int V = 6;

void initAdjMatrix(int adjMatrix[V][V]) {

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

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

adjMatrix[i][j] = 0;

}

}

void addEdge(int adjMatrix[V][V], int u, int v) {

adjMatrix[u][v] = 1;

adjMatrix[v][u] = 1;

}

void printAdjMatrix(int adjMatrix[V][V]) {

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

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

cout << adjMatrix[i][j] << " ";

}

cout << endl;

}

}

int main() {

int adjMatrix[V][V];

initAdjMatrix(adjMatrix);

addEdge(adjMatrix, 0, 3);

addEdge(adjMatrix, 0, 4);

addEdge(adjMatrix, 1, 4);

addEdge(adjMatrix, 1, 5);

addEdge(adjMatrix, 1, 2);

addEdge(adjMatrix, 2, 5);

addEdge(adjMatrix, 2, 3);

addEdge(adjMatrix, 3, 5);

addEdge(adjMatrix, 4, 5);

cout << "Adjacency Matrix:" << endl;

printAdjMatrix(adjMatrix);

return 0;

}

**13. Write a c++ program to sort a given array using bubble sort.**

**Ans :-**

#include<iostream>

using namespace std;

void bubbleSort(int arr[],int n){

for(int i = 0;i < n-1;i++){

for(int j = 0;j < n-i-1;j++){

if(arr[j]>arr[j+1]){

swap(arr[j],arr[j+1]);

}

}

}

}

void printArray(int arr[],int n){

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main(){

int n = 6;

int arr[] = {6,5,4,9,8,2};

bubbleSort(arr,n);

printArray(arr,n);

return 0;

}

**14. Write a c++ program to sort a given array using merge sort.**

**Ans :-**

#include<iostream>

using namespace std;

void bubbleSort(int arr[],int n){

for(int i = 0;i < n-1;i++){

for(int j = 0;j < n-i-1;j++){

if(arr[j]>arr[j+1]){

swap(arr[j],arr[j+1]);

}

}

}

}

void printArray(int arr[],int n){

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main(){

int n = 6;

int arr[] = {6,5,4,9,8,2};

bubbleSort(arr,n);

printArray(arr,n);

return 0;

}

**15. Write a c++ program that Implements a stack and its functions, such as push, pop, isEmpty, top, size.**

**Ans :-**

#include <iostream>

using namespace std;

int stack[5];

int top = -1;

void push(int val) {

if (top < 4) {

stack[++top] = val;

} else {

cout << "Stack Overflow" << endl;

}

}

void pop() {

if (top >= 0) {

cout << "Popped element: " << stack[top--] << endl;

} else {

cout << "Stack Underflow" << endl;

}

}

void display() {

if (top >= 0) {

cout << "Stack elements: ";

for (int i = top; i >= 0; i--) {

cout << stack[i] << " ";

}

cout << endl;

} else {

cout << "Stack is empty" << endl;

}

}

int main() {

int choice, value;

do {

cout << "1) Push\n2) Pop\n3) Display\n4) Exit\nEnter choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to push: ";

cin >> value;

push(value);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

cout << "Exiting..." << endl;

break;

default:

cout << "Invalid choice" << endl;

}

} while (choice != 4);

return 0;

}

**16. Write a c++ program that implements a queue and its functions, such as enqueue, dequeue, front, isEmpty, size.**

**Ans :-**

#include <iostream>

using namespace std;

int queue[100], front = -1, rear = -1;

void Insert() {

int val;

if (rear == 99) {

cout << "Queue Overflow" << endl;

} else {

if (front == -1) front = 0;

cout << "Insert element: ";

cin >> val;

queue[++rear] = val;

}

}

void Delete() {

if (front == -1 || front > rear) {

cout << "Queue Underflow" << endl;

} else {

cout << "Deleted: " << queue[front++] << endl;

}

}

void Display() {

if (front == -1 || front > rear) {

cout << "Queue is empty" << endl;

} else {

cout << "Queue elements: ";

for (int i = front; i <= rear; i++) {

cout << queue[i] << " ";

}

cout << endl;

}

}

bool isEmpty() {

return front == -1 || front > rear;

}

int size() {

return isEmpty() ? 0 : rear - front + 1;

}

int frontElement() {

return isEmpty() ? -1 : queue[front];

}

int main() {

int choice;

do {

cout << "\n1) Insert\n2) Delete\n3) Display\n4) Check if empty\n5) Get size\n6) Front element\n7) Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: Insert(); break;

case 2: Delete(); break;

case 3: Display(); break;

case 4: cout << (isEmpty() ? "Queue is empty" : "Queue is not empty") << endl; break;

case 5: cout << "Size: " << size() << endl; break;

case 6: cout << (frontElement() == -1 ? "Queue is empty" : "Front element: " + to\_string(frontElement())) << endl; break;

case 7: cout << "Exit" << endl; break;

default: cout << "Invalid choice" << endl;

}

} while (choice != 7);

return 0;

}

**17. Write a c++ function that implements topological sort.**

**Ans :-**

#include <iostream>

#include <vector>

#include <stack>

using namespace std;

void addEdge(vector<vector<int>> &adj, int u, int v) {

adj[u].push\_back(v);}

void dfs(int node, vector<vector<int>> &adj, vector<bool> &visited, stack<int> &st) {

visited[node] = true;

for (int neighbor : adj[node]) {

if (!visited[neighbor]) {

dfs(neighbor, adj, visited, st); } }

st.push(node);}

vector<int> topoSort(int V, vector<vector<int>> &adj) {

stack<int> st;

vector<bool> visited(V, false);

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

if (!visited[i]) {

dfs(i, adj, visited, st); }

}

vector<int> result;

while (!st.empty()) {

result.push\_back(st.top());

st.pop(); }

return result;

}

int main() {

int V = 6;

vector<vector<int>> adj(V);

addEdge(adj, 5, 0);

addEdge(adj, 5, 2);

addEdge(adj, 4, 0);

addEdge(adj, 4, 1);

addEdge(adj, 2, 3);

addEdge(adj, 3, 1);

vector<int> result = topoSort(V, adj);

for (int i : result) {

cout << i << " "; }

cout << endl;

return 0;

}

**18. Write a c++ program that implements Binary Search.**

**Ans :-**

#include<iostream>

using namespace std;

void SelectionSort(int arr[],int n){

for(int i = 0;i<n-1;i++){

int smallestindex = i;

for(int j = i+1;j<n;j++){

if(arr[j]<arr[smallestindex]){

smallestindex = j;

}

}

swap(arr[i],arr[smallestindex]);

}

}

void printArray(int arr[],int n){

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main(){

int n = 6;

int arr[] = {6,5,4,9,8,2};

SelectionSort(arr,n);

printArray(arr,n);

return 0;

}

**19. Write a c++ program that implements selection sort.**

**Ans :-**

#include <bits/stdc++.h>

using namespace std;

int binarySearch(vector<int> &nums, int key, int low, int high) {

while (low <= high) {

int mid = low + (high - low) / 2;

if (nums[mid] == key) return mid;

else if (nums[mid] < key) low = mid + 1;

else high = mid - 1;

}

return -1;

}

int main() {

vector<int> nums = {10, 50, 30, 70, 80, 20, 90, 40};

sort(nums.begin(), nums.end());

cout << "After sorting the array becomes: ";

for (int i : nums) cout << i << " ";

cout << endl;

int res = binarySearch(nums, 40, 0, nums.size() - 1);

if (res == -1) {

cout << "Key not found!!" << endl;

} else {

cout << "Key found and the index is " << res << endl;

}

return 0;

}

**20. Write a c++ program to store a directed graph using adjacency List/ adjacency matrix (any one method).**

**Ans :-**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int V = 5;

vector<vector<int>> adjList(V);

adjList[0].push\_back(1);

adjList[0].push\_back(2);

adjList[1].push\_back(3);

adjList[2].push\_back(3);

adjList[3].push\_back(4);

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

cout << "Vertex " << i << " -> ";

for (int j : adjList[i]) {

cout << j << " ";

}

cout << endl;

}

return 0;

}