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;

}

**3. 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>

using namespace std;

struct Node

{

int PRN;

string name;

int age;

Node\*next;

};

int main()

{

Node\*first = new Node{2046,"Aditya",19};

Node\*second = new Node{2054,"yash",20};

first -> next = second;

Node\*temp = first;

while (temp)

{

cout<<"PRN:" <<temp->PRN <<"Name:" <<temp->name <<"Age:" <<temp->age <<endl;

temp=temp->next;

}

delete first;

delete second;

return 0;

}

**4. 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> RollNumber;

RollNumber.push\_back(46);

RollNumber.push\_back(55);

RollNumber.push\_back(89);

RollNumber.push\_back(24);

RollNumber.push\_back(78);

cout << "Roll Numbers in the vector: " <<RollNumber.size() endl;

return 0;

}

**5. 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 val) {

data = val;

left = nullptr;

right = nullptr;

}

};

void inOrder(node\* root) {

if (root) {

inOrder(root->left);

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

inOrder(root->right);

}

}

int main() {

node\* root = new node(15);

root->left = new node(10);

root->right = new node(20);

root->left->right = new node(11);

cout << "Binary tree constructed using In-order: " ;

inOrder(root);

cout << endl;

return 0;

}

**7. Write a c++ program for construction of a Binary Search Tree. (use in-order)**

**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 inorder(Node\* root)

{

if (root) {

inorder(root->left);

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

inorder(root->right);

}

}

void preorder(Node\* root) {

if (root) {

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

preorder(root->left);

preorder(root->right);

}

}

void postorder(Node\* root) {

if (root) {

postorder(root->left);

postorder(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: ";

inorder(root);

cout << endl;

cout << "Preorder Traversal: ";

preorder(root);

cout << endl;

cout << "Postorder Traversal: ";

postorder(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>

using namespace std;

int main()

{

int vertices, edges, u, v;

cin>>vertices>>edges;

vector<int>adj[vertices];

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

{

cin>>u>>v;

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

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

}

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

{

cout<<i<<" ";

for(int neighbour : adj[i])

{

cout<<neighbour<<" ";

}

cout<<endl;

}

return 0;

}

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

**Ans :-**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int vertices, edges, u, v;

cin >> vertices >> edges;

vector<int> adj[vertices];

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

cin >> u >> v;

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

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

}

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

cout << i << ": ";

for (int neighbour : adj[i]) {

cout << neighbour << " ";

}

cout << endl;

}

return 0;

}

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

**Ans :-**

#include <iostream>

using namespace std;

int main() {

int i, j, n, temp;

cout << "Enter size of array: ";

cin >> n;

int a[n];

cout << "Enter array elements: ";

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

cin >> a[i];

}

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

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

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

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

cout << "Sorted array: ";

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

cout << a[i] << " ";

}

cout << endl;

return 0;

}

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

**Ans :-**

#include <bits/stdc++.h>

using namespace std;

void merge(vector<int>& arr, int left, int mid, int right)

{

int n1 = mid - left + 1;

int n2 = right - mid;

vector<int> L(n1), R(n2);

for (int i = 0; i < n1; i++)

L[i] = arr[left + i];

for (int j = 0; j < n2; j++)

R[j] = arr[mid + 1 + j];

int i = 0, j = 0;

int k = left;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(vector<int>& arr, int left, int right)

{

if (left >= right)

return;

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

void printVector(vector<int>& arr)

{

for (int i = 0; i < arr.size(); i++)

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

cout << endl;

}

int main()

{

vector<int> arr = { 12, 11, 13, 5, 6, 7 };

int n = arr.size();

cout << "Given vector is \n";

printVector(arr);

mergeSort(arr, 0, n - 1);

cout << "\nSorted vector is \n";

printVector(arr);

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;

int main() {

int i, n, s, mid, e, flag = 0, key;

cout << "Enter size: ";

cin >> n;

int arr[n];

s = 0;

e = n - 1;

cout << "Enter elements: ";

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

cin >> arr[i];

}

cout << "Enter key element: ";

cin >> key;

while (s <= e) {

mid = (s + e) / 2;

if (arr[mid] == key) {

flag = 1;

cout << "Element found!" << endl;

break;

} else if (arr[mid] > key) {

e = mid - 1;

} else {

s = mid + 1;

}

}

if (flag == 0) {

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

}

return 0;

}

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

**Ans :-**

#include <iostream>

using namespace std;

void SelectionSort(int arr[], int n)

{

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

{

int midindex = i;

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

{

if(arr[j] < arr[midindex])

{

swap(arr[j], arr[midindex]);

}

}

}

}

void printdata(int arr[], int n)

{

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

{

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

}

}

int main() {

int n;

cout << "Enter the number of elements in the array: ";

cin >> n;

int arr[n];

cout << "Enter the elements of the array: ";

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

{

cin >> arr[i];

}

cout << "Original array: ";

printdata(arr, n);

SelectionSort(arr, n);

cout << "Sorted array: ";

printdata(arr, n);

return 0;

}