**2. B) C++ program to implement iterative preorder traversal**

**#include <bits/stdc++.h>**

**using namespace std;**

**/\* A binary tree node has data, left child and right child \*/**

**struct node {**

**int data;**

**struct node\* left;**

**struct node\* right;**

**};**

**struct node\* newNode(int data)**

**{**

**struct node\* node = new struct node;**

**node->data = data;**

**node->left = NULL;**

**node->right = NULL;**

**return (node);**

**}**

**// An iterative process to print preorder traversal of Binary tree**

**void iterativePreorder(node\* root)**

**{**

**if (root == NULL)**

**return;**

**// Create an empty stack and push root to it**

**stack<node\*> nodeStack;**

**nodeStack.push(root);**

**while (nodeStack.empty() == false) {**

**// Pop the top item from stack and print it**

**struct node\* node = nodeStack.top();**

**printf("%d ", node->data);**

**nodeStack.pop();**

**// Push right and left children of the popped node to stack**

**if (node->right)**

**nodeStack.push(node->right);**

**if (node->left)**

**nodeStack.push(node->left);**

**}**

**}**

**int main()**

**{**

**struct node\* root = newNode(10);**

**root->left = newNode(8);**

**root->right = newNode(2);**

**root->left->left = newNode(3);**

**root->left->right = newNode(5);**

**root->right->left = newNode(2);**

**iterativePreorder(root);**

**return 0;**

**}**

**Output**

**10 8 3 5 2 2**

**2. c) C++ program to implement iterative postorder traversal**

**#include <bits/stdc++.h>**

**using namespace std;**

**// A tree node**

**struct Node {**

**int data;**

**Node \*left, \*right;**

**};**

**// Function to create a new node with the given data**

**Node\* newNode(int data)**

**{**

**Node\* node = new Node;**

**node->data = data;**

**node->left = node->right = NULL;**

**return node;**

**}**

**// An iterative function to do post order**

**// traversal of a given binary tree**

**void postOrderIterative(Node\* root)**

**{**

**if (root == NULL)**

**return;**

**// Create two stacks**

**stack<Node \*> s1, s2;**

**// push root to first stack**

**s1.push(root);**

**Node\* node;**

**// Run while first stack is not empty**

**while (!s1.empty()) {**

**// Pop an item from s1 and push it to s2**

**node = s1.top();**

**s1.pop();**

**s2.push(node);**

**// Push left and right children**

**// of removed item to s1**

**if (node->left)**

**s1.push(node->left);**

**if (node->right)**

**s1.push(node->right);**

**}**

**// Print all elements of second stack**

**while (!s2.empty()) {**

**node = s2.top();**

**s2.pop();**

**cout << node->data << " ";**

**}**

**}**

**// Driver code**

**int main()**

**{**

**// Let us construct the tree**

**// shown in above figure**

**Node\* root = NULL;**

**root = newNode(1);**

**root->left = newNode(2);**

**root->right = newNode(3);**

**root->left->left = newNode(4);**

**root->left->right = newNode(5);**

**root->right->left = newNode(6);**

**root->right->right = newNode(7);**

**postOrderIterative(root);**

**return 0;**

**}**

**Output:**

**4 5 2 6 7 3 1**

**2. d) To find fibonacci series using iteration.**

**#include <iostream>**

**using namespace std;**

**void fib(int num) {**

**int x = 0, y = 1, z = 0;**

**for (int i = 0; i < num; i++) {**

**cout << x << " ";**

**z = x + y;**

**x = y;**

**y = z;**

**}**

**}**

**int main() {**

**int num;**

**cout << "Enter the number : ";**

**cin >> num;**

**cout << "\nThe fibonacci series : " ;**

**fib(num);**

**return 0;**

**}**

**Output**

**Enter the number : 10**

**The fibonacci series : 0 1 1 2 3 5 8 13 21 34**