2. Implementation of iteration functions for tree traversal and Fibonacci.

2(a). Inorder

#include <iostream>

#include <stack>

using namespace std;

struct Node

{

    int data;

    struct Node\* left;

    struct Node\* right;

    Node (int data)

    {

        this->data = data;

        left = right = NULL;

    }

};

void inOrderTraversal(Node \*root)

{

*// Create an empty stack.*

    stack<Node\*> stack;

*// Start from the root node.*

    Node \*curr = root;

*// If the current node is null and stack is also empty, the algorithm terminates.*

    while (!stack.empty() || curr != NULL)

    {

*// Push the current node to the stack and set current=current->left.*

        if (curr != NULL)

        {

            stack.push(curr);

            curr = curr->left;

        }

        else *// If the curr node is NULL.*

        {

            curr = stack.top();

            stack.pop(); *// Pop the node on top of stack.*

            cout << curr->data << " "; *// Print it.*

            curr = curr->right;

        }

    }

}

*// Driver code*

int main() {

    struct 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);

    inOrderTraversal(root);

  return 0;

}

Output

4 2 5 1 3