**1.Write a C Program to Sort a stack using a temporary stack**

#include <bits/stdc++.h>

using namespace std;

stack<int> sortStack(stack<int> &input) {

stack<int> tmpStack;

while (!input.empty()) {

int tmp = input.top();

input.pop();

while (!tmpStack.empty() && tmpStack.top() > tmp)

{

input.push(tmpStack.top());

tmpStack.pop();

}

tmpStack.push(tmp);

}

return tmpStack;

}

int main()

{

stack<int> input;

input.push(34);

input.push(3);

input.push(31);

input.push(98);

input.push(92);

input.push(23);

stack<int> tmpStack = sortStack(input);

cout << "Sorted numbers are:\n";

while (!tmpStack.empty())

{

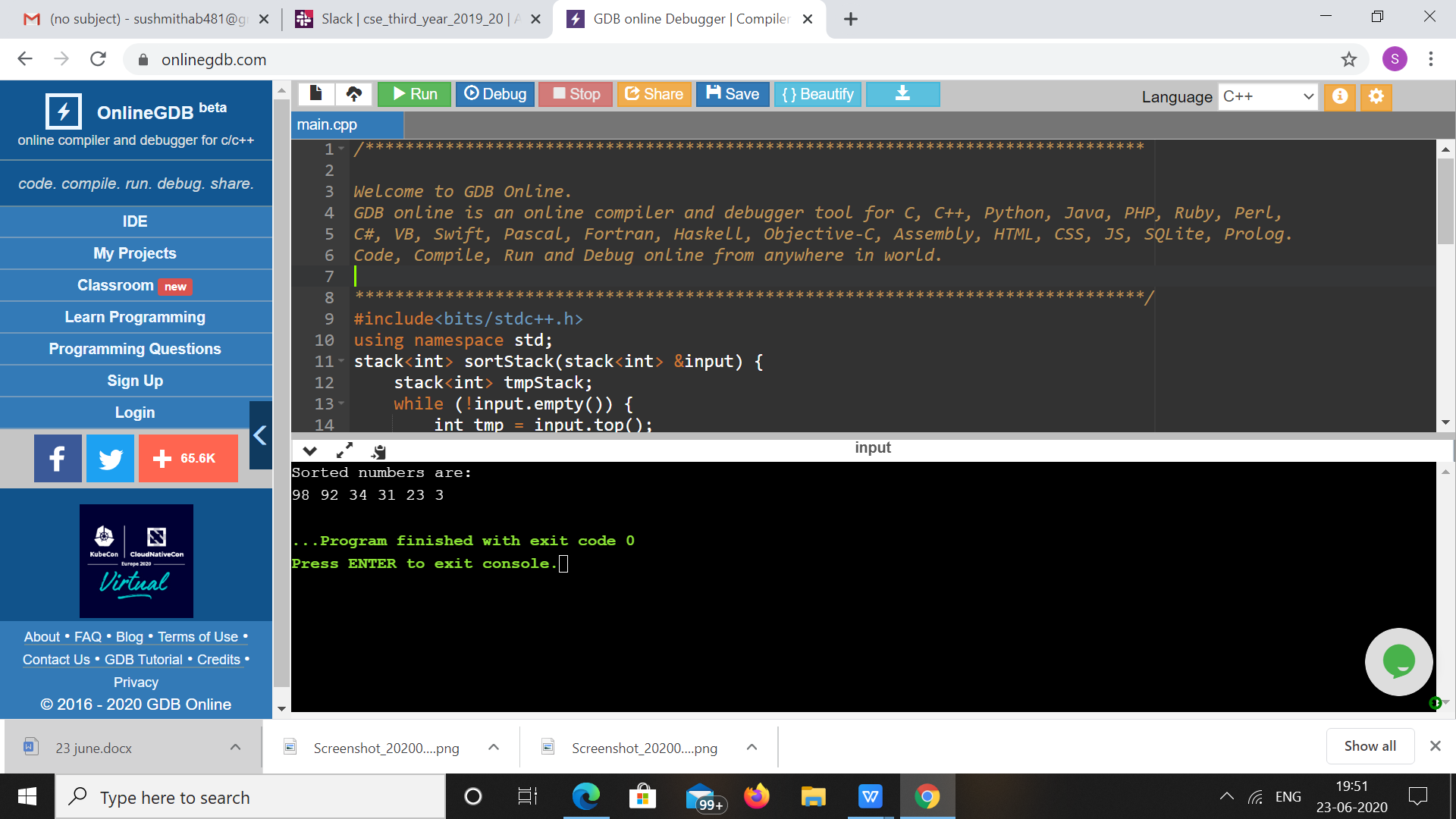
cout << tmpStack.top()<< " ";

tmpStack.pop();

}

}

**OUTPUT**

****

**2.Write a Java Program to traverse a binary tree using PreOrder traversal without recursion**

import java.util.Stack;

public class Main {

public static void main(String[] args) throws Exception {

BinaryTree bt = BinaryTree.create();

 System.out .println("printing nodes of a binary tree in preOrder using recursion");

bt.preOrderWithoutRecursion();

}

}

class BinaryTree {

static class TreeNode {

String data;

TreeNode left, right;

TreeNode(String value) {

 this.data = value; left = right = null;

}

boolean isLeaf() {

return left == null ? right == null : false;

} }

TreeNode root;

 public void preOrderWithoutRecursion() {

Stack<TreeNode> nodes = new Stack<>();

nodes.push(root);

while (!nodes.isEmpty()) {

TreeNode current = nodes.pop();

 System.out.printf("%s ", current.data);

 if (current.right != null) {

nodes.push(current.right);

}

if (current.left != null) {

nodes.push(current.left);

} } }

public static BinaryTree create() {

BinaryTree tree = new BinaryTree();

TreeNode root = new TreeNode("a");

tree.root = root;

tree.root.left = new TreeNode("b");

tree.root.left.left = new TreeNode("c");

tree.root.left.right = new TreeNode("d");

tree.root.right = new TreeNode("e");

tree.root.right.right = new TreeNode("f");

return tree;

} }

**OUTPUT**

