Muzamil Sikander   
CS221226

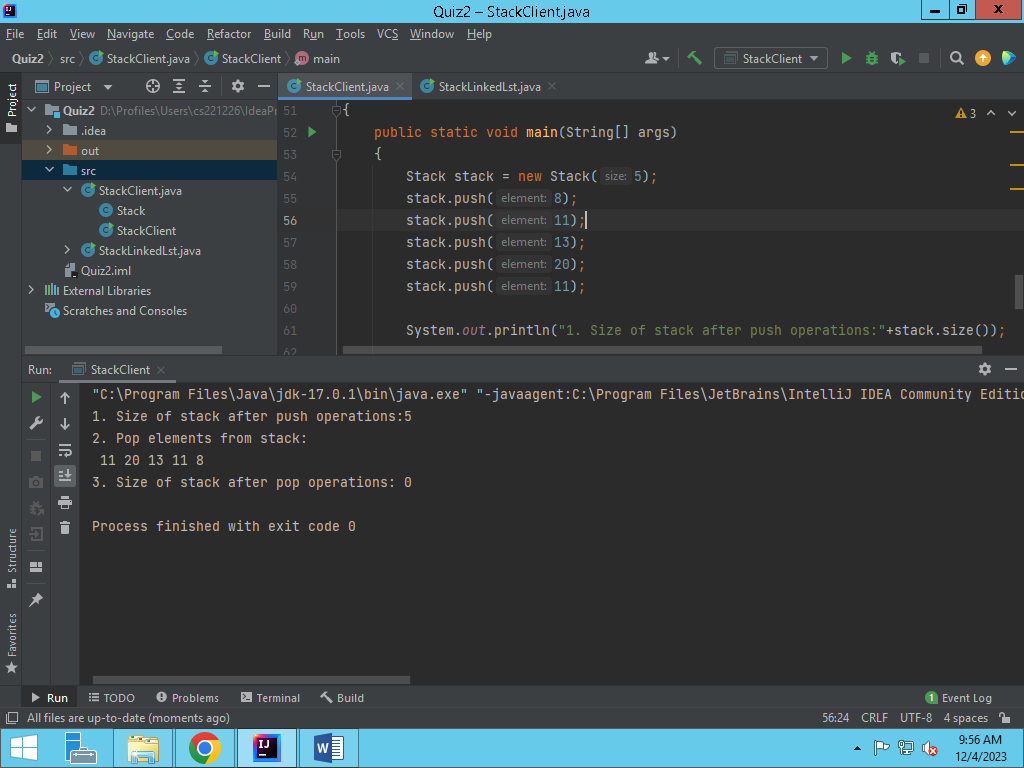
DATA STRUCTURE AND

ALGORTHEM

QUIZ NO 2

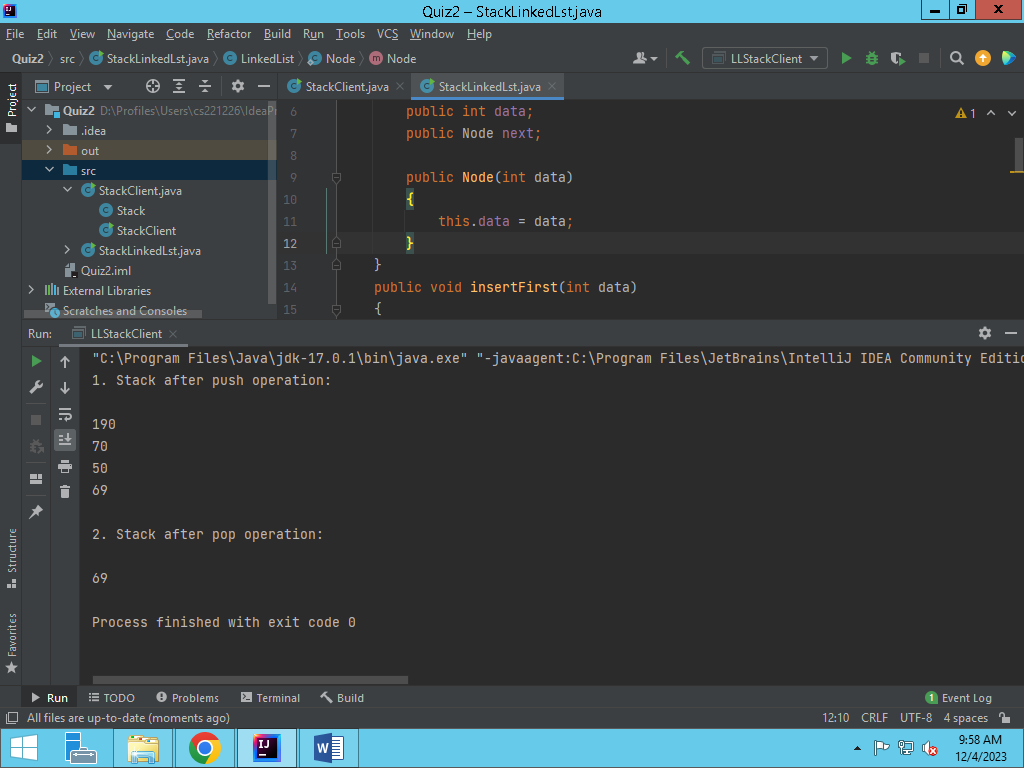
QUESTION 1 CODE)

class Stack  
{  
 int arr[];  
 int size;  
 int index = 0;  
  
 Stack(int size)  
 {  
 this.size = size;  
 arr = new int[size];  
 }  
 public boolean isEmpty()  
 {  
 if(index == 0)  
 {  
 return true;  
 }  
 return false;  
 }  
 public boolean isFull()  
 {  
 if(index == size)  
 {  
 return true;  
 }  
 return false;  
 }  
 public void push(int element)  
 {  
 if(isFull())  
 {  
 System.*out*.println("Stack is Full!");  
 }  
 arr[index] = element;  
 index++;  
 }  
 public int pop()  
 {  
 if(isEmpty())  
 {  
 System.*out*.println("Stack is Empty!");  
 }  
 return arr[--index];  
 }  
 public int size()  
 {  
 return index;  
 }  
}  
class StackClient  
{  
 public static void main(String[] args)  
 {  
 Stack stack = new Stack(5);  
 stack.push(8);  
 stack.push(11);  
 stack.push(13);  
 stack.push(20);  
 stack.push(11);  
  
 System.*out*.println("1. Size of stack after push operations:"+stack.size());  
  
 System.*out*.println("2. Pop elements from stack: ");  
 while(!stack.isEmpty())  
 {  
 System.*out*.printf(" %d",stack.pop());  
 }  
 System.*out*.println("\n3. Size of stack after pop operations:" +  
 " "+stack.size());  
 }  
}

QUESTION 1 OUTPUT) 

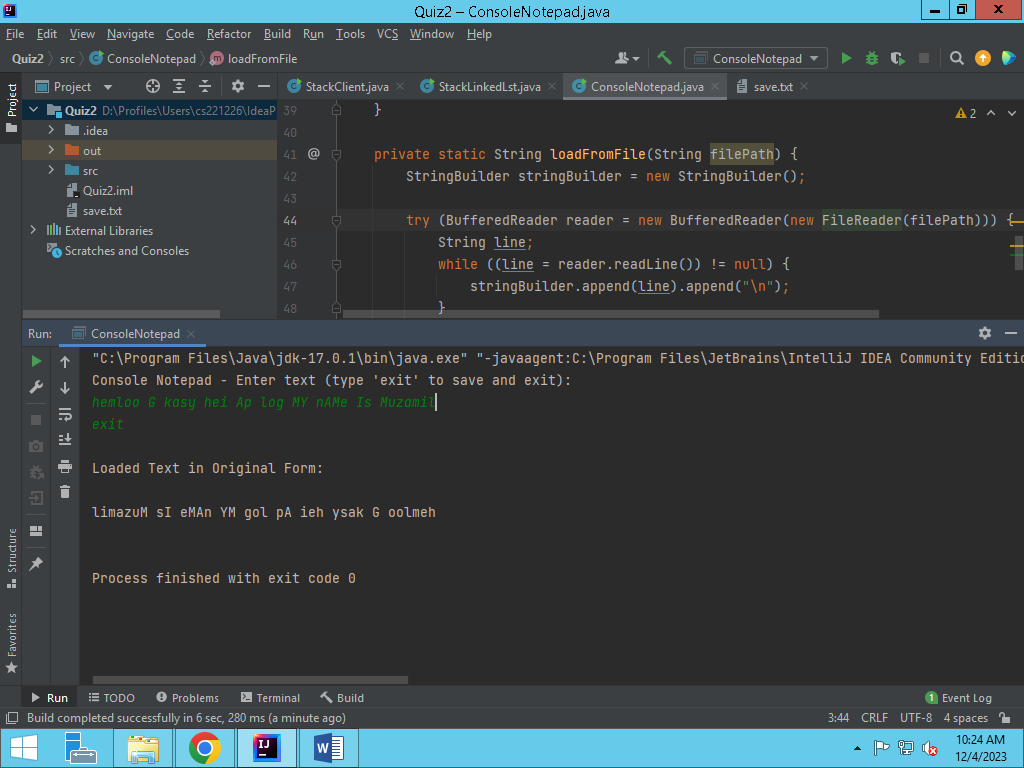
QUESTION 2 CODE )

class LinkedList  
{  
 Node top;  
 public static class Node  
 {  
 public int data;  
 public Node next;  
  
 public Node(int data)  
 {  
 this.data = data;  
 }  
 }  
 public void insertFirst(int data)  
 {  
 Node n = new Node(data);  
 n.next = top;  
 top = n;  
 }  
 public Node deleteFirst()  
 {  
 Node temp = top;  
 top = top.next;  
 return temp;  
 }  
 public void displayList()  
 {  
 Node curr = top;  
 while(curr != null)  
 {  
 System.*out*.println(curr.data);  
 curr = curr.next;  
 }  
 }  
}  
class LinkedListStack  
{  
 LinkedList l1 = new LinkedList();  
  
 public void push(int element)  
 {  
 l1.insertFirst(element);  
 }  
 public void pop()  
 {  
 l1.deleteFirst();  
 }  
 public void displayStack()  
 {  
 System.*out*.println(" ");  
 l1.displayList();  
 }  
}  
class LLStackClient  
{  
 public static void main(String[] args) {  
 LinkedListStack s1 = new LinkedListStack();  
  
  
 s1.push(69);  
 s1.push(50);  
 s1.push(70);  
 s1.push(190);  
  
  
 System.*out*.println("1. Stack after push operation: ");  
 s1.displayStack();  
 System.*out*.println("\n2. Stack after pop operation: ");  
 s1.pop();  
 s1.pop();  
 s1.pop();  
 s1.displayStack();  
  
 }  
}

QUESTION 2 OUTPUT

QUESTION 3 CODE)

import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class ConsoleNotepad {  
 public static void main(String[] args) {  
  
 System.*out*.println("Console Notepad - Enter text (type 'exit' to save and exit):");  
 String inputText = *getUserInput*();  
  
 String reversedText = *reverseText*(inputText);  
 *saveToFile*("save.txt", reversedText);  
  
 String originalText = *loadFromFile*("save.txt");  
 System.*out*.println("\nLoaded Text in Original Form: \n" + originalText);  
 }  
  
 private static String getUserInput() {  
 StringBuilder userInput = new StringBuilder();  
 try (BufferedReader reader = new BufferedReader(new java.io.InputStreamReader(System.*in*))) {  
 String line;  
 while (!(line = reader.readLine()).equalsIgnoreCase("exit")) {  
 userInput.append(line).append("\n");  
 }  
 } catch (IOException e) {  
 throw new RuntimeException("Error reading user input: " + e.getMessage());  
 }  
 return userInput.toString();  
 }  
  
 private static void saveToFile(String filePath, String text) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {  
 writer.write(text);  
 } catch (IOException e) {  
 throw new RuntimeException("Error saving to file: " + e.getMessage());  
 }  
 }  
  
 private static String loadFromFile(String filePath) {  
 StringBuilder stringBuilder = new StringBuilder();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 stringBuilder.append(line).append("\n");  
 }  
 } catch (IOException e) {  
 throw new RuntimeException("Error loading from file: " + e.getMessage());  
 }  
  
 return stringBuilder.toString();  
 }  
  
 private static String reverseText(String text) {  
 return new StringBuilder(text).reverse().toString();  
 }  
}

QUESTION 3 OUTPUT)