**ASSIGNMENT 03**

**TASK 01**

|  |
| --- |
| **CODE**  package task01;  import java.util.\*;  public class Task01 {  // Pushes integer values in stack  static void push(int data) {  push(data);  }  // Pop out Stack peak data  static Integer pop() {  return pop();  }  // Inserts Stack T data in Stack S recursively  static Stack<Integer> mergeStacks(Stack<Integer> S, Stack<Integer> T){  // Check whether T stack is empty or not  if(T.size()<=0){  return S;  }    int data = T.pop();  mergeStacks(S,T);  S.push(data);    return S;  }  public static void main(String[] args) {  // Creates 3 stacks R,S & T  Stack<Integer> R = new Stack<Integer>();  Stack<Integer> S = new Stack<Integer>();  Stack<Integer> T = new Stack<Integer>();    // Pushes data in stack R  R.push(1);  R.push(2);  R.push(3);    // Pushes data in stack S  S.push(4);  S.push(5);  // Pushes data in stack T  T.push(6);  T.push(7);  T.push(8);  T.push(9);  System.out.println("Stack Elements Before Configuration: ");    System.out.println("\tStack R: "+R.toString());  System.out.println("\tStack S: "+S.toString());  System.out.println("\tStack T: "+T.toString()+"\n");  // Stores stack S data in an array  List<Integer> s\_data = new ArrayList<>();  for (int i = 0; i <= S.size(); i++) {  s\_data.add(S.pop());  }  // mergeStacks method is being called  mergeStacks(S,T);    // stack s original data is pushed back into stack s  for (int i=s\_data.size()-1; i>=0; i--){  S.push(s\_data.get(i));  }    System.out.println("Stack Elements After Configuration: ");    System.out.println("\tStack R: "+R.toString());  System.out.println("\tStack S: "+S.toString());  System.out.println("\tStack T: "+T.toString()+"\n");    }  } |

|  |
| --- |
| **SCREENSHOT** |

**TASK 02**

|  |
| --- |
| **CODE**  package task02;  // Libraries to import  import java.util.\*;  import static task02.Search.stack;  public class Task02 {  public static void main(String[] args) {  //Input taken from user for searching purpose  Scanner sc = new Scanner(System.in);  System.out.print("Enter an integer: ");  int x = sc.nextInt();  Search s = new Search();  System.out.println("\nStack before searching: " + stack.toString());  //Method is called which returns the positions of element in stack if found.  System.out.println("\n/\*\* Position Starts from 0... \*\*/");  System.out.println(x + " found at position: " + s.searchElement(x));  System.out.println("\nStack after searching: " + stack.toString());  }  }  //Implementation of Search Class  class Search {  //Array with initial values to be inserted in stack  Integer[] stackItems = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};  int size = 0; //Size of stack  //Stack and Queue objects are created  static Stack<Integer> stack = new Stack<>();  static Queue<Integer> queue = new LinkedList<>();  //Constructor declaration  Search() {  //Pushing elements of stackItems in stack  for (int i : stackItems) {  stack.push(i);  size++;  }  }  //This method searches position of element in stack using queue  Integer searchElement(int varToSearch) {  int temp;  int position = -1;  //pops item by items from stack, compare with varToSearch and pushes the items in queue  for (int i = 0; i < size; i++) {  temp = stack.pop();  if (varToSearch == temp) {  position = i;  }  queue.add(temp);  }  //Items are pushed back from queue to Stack  stackInsert();  //Position of element (From Bottom to Top)in stack is returned  return stack.size() - position - 1;  }  //Pushes queue items back in stack through recursion  void stackInsert() {  if (size <= 0) {  return;  }  int v = -1;  v = queue.remove();  size--;  stackInsert();  stack.push(v);  }  } |

|  |
| --- |
| **SCREENSHOT** |

**TASK 03**

|  |
| --- |
| **CODE**  package task03;  //Libraries to import  import java.util.\*;  public class Task03 {  public static void main(String[] args) {  //Objects of Bob and Class are made  Bob b = new Bob();  Alice a = new Alice();  //Message to be sent by Bob to Alice. I have taken it as a string  String str = "Hi Alice! I am Bob. I want to send you a message.";  System.out.println("Sending Message: " + str);    //String is splitted into words. Each word represents 1 packet  String[] arr = str.split(" ");  int count = 0; //counts the number of packets  for (String s : arr) {  b.makePacket(count, s); //makePacket() method is called which will make a hashmap of packet number and packets  count++;  }  b.sendPacket(); //Sends packet 1 by 1 to Alice  System.out.println("\nReceived Message: " + a.combinePackets()); //Outputs the message received by Alice  }  }  //Bob Side Working  class Bob {  //Packets to be sent are stored in this buffer(HashMap)  Map<Integer, String> dataPackets = new HashMap<>();  //Make Packets by inserting packet data and seq num to hashmap  void makePacket(int index, String packet\_data) {  dataPackets.put(index, packet\_data);  }  //Sned the packets to Alice Computer by creating its object  void sendPacket() {  Alice a = new Alice();  String data;  for (int i = 0; i < dataPackets.size(); i++) {  data = dataPackets.get(i);  System.out.println("\nSending: " + "packet " + i + "...");  a.recPacket(i, data);  }  }  }  //Alice Side Working  class Alice {  //Received Packets are stored in this hashMap  static Map<Integer, String> dataPackets = new HashMap<>();    //Packets sent by Bob are received here and stored in Buffer(Hashmap)  //If packets are not in order then they are automatically ordered by HashMap  void recPacket(Integer index, String packet) {  dataPackets.put(index, packet);  System.out.println("Receiving: " + "packet " + index + "...");  }  //Combines the packets and creates a complete message  String combinePackets() {  String message = "";  for (String s : dataPackets.values()) {  message += s + " ";  }  return message;  }  } |

|  |
| --- |
| **SCREENSHOT** |