CSC248
FUNDAMENTALS OF DATA STRUCTURE

LAB ASSIGNMENT 5

NAME	: MUHAMMAD REDZA BIN MAHAYADIN
STUDENT ID	: 2022676696
GROUP	: RCDCS1103B
LECTURER	: SIR MOHD NIZAM BIN OSMAN

QUESTION 1

```java
import java.util.Stack;
import java.util.Scanner;

public class DecToHex {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Stack<Integer> stack = new Stack<Integer>();

        System.out.print("Enter a decimal number: ");
        int dec = in.nextInt();
        int dec2 = dec;
        int quo = 0;
        int rem = 0;
        String hex = "";

        System.out.printf("¥n%10s  %3s¥n", "QUO", "REM");
        System.out.printf("%10s, %3s¥n", dec, "---");
        while (dec > 0) {
            quo = dec / 16;
            rem = dec % 16;
            dec = quo;
            stack.push(rem);
            System.out.printf("%10d, %3d¥n", quo, stack.peek());
        }

        while (!stack.isEmpty()) {
            int num = stack.pop();
            if (num < 10) {
                hex += num;
            } else {
                switch (num) {
                    case 10:
                        hex += "A";
                        break;
                    case 11:
                        hex += "B";
                        break;
                    case 12:
                        hex += "C";
                        break;
                    case 13:
                        hex += "D";
                        break;
                    case 14:
                        hex += "E";
                        break;
                    case 15:
                        hex += "F";
                        break;
                }
            }
        }
        System.out.println("¥nDecimal    : " + dec2);
        System.out.println("Hexadecimal: " + hex);


    }
}
```

QUESTION 2

```java
import java.util.Scanner;

public class PostfixApp {
    public static void main(String[] args) {
        Stack stack = new Stack();
        Scanner in = new Scanner(System.in);

        System.out.println("Enter postfix expression: ");
        String expression = in.nextLine();

        String[] tokens = expression.split(" ");
        for (String token : tokens) {
            try {
                int num = Integer.parseInt(token);
                stack.push(num);
            } catch (NumberFormatException e) {
                int op2 = (int) stack.pop();
                int op1 = (int) stack.pop();

                switch (token) {
                    case "+":
                        stack.push(op1 + op2);
                        break;
                    case "-":
                        stack.push(op1 - op2);
                        break;
                    case "*":
                        stack.push(op1 * op2);
                        break;
                    case "/":
                        stack.push(op1 / op2);
                        break;
                }
            }
        }

        int result = (int) stack.pop();
        System.out.println("Result: " + result);
        in.close();
    }
}


class ListNode {
    private Object data;
    private ListNode next;

    public ListNode() {
        this(null, null);
    }

    public ListNode(Object obj) {
        this(obj, null);
    }

    public ListNode(Object obj, ListNode node) {
        this.data = obj;
        this.next = node;
    }

    public Object getData() {
        return this.data;
    }

    public void setData(Object obj) {
        this.data = obj;
    }

    public ListNode getNext() {
        return this.next;
    }

    public void setNext(ListNode node) {
        this.next = node;
    }
}

class List {
    private ListNode firstNode;
    private ListNode lastNode;
    private String name;

    public List() {
        this("list");
    }

    public List(String ListName) {
        name = ListName;
        firstNode = lastNode = null;
    }
    public ListNode getFirstNode() {
        return this.firstNode;
    }

    public void setFirstNode(ListNode firstNode) {
        this.firstNode = firstNode;
    }

    public ListNode getLastNode() {
        return this.lastNode;
    }

    public void setLastNode(ListNode lastNode) {
        this.lastNode = lastNode;
    }

    public void insertAtFront(Object insertItem) {
        if (isEmpty()) {
            firstNode = lastNode = new ListNode(insertItem);
        } else {
            firstNode = new ListNode(insertItem, firstNode);
        }
    }

    public Object removeFromFront() throws EmptyListException {
        if (isEmpty()) {
            throw new EmptyListException(name);
        }

        Object removedItem = firstNode.getData();

        if (firstNode == lastNode) {
            firstNode = lastNode = null;
        } else {
            firstNode = firstNode.getNext();
        }

        return removedItem;
    }

    public boolean isEmpty() {
        return firstNode == null;
    }
}
```

```
class Stack extends List {
    public Stack() {
    }

    public void push(Object obj) {
        this.insertAtFront(obj);
    }

    public Object pop() {
        return this.removeFromFront();
    }

    public Object peek() {
        return this.getFirstNode().getData();
    }
}

class EmptyListException extends RuntimeException {
    public EmptyListException() {
        this("List");
    }

    public EmptyListException(String name) {
        super(name + " is empty");
    }
}
```