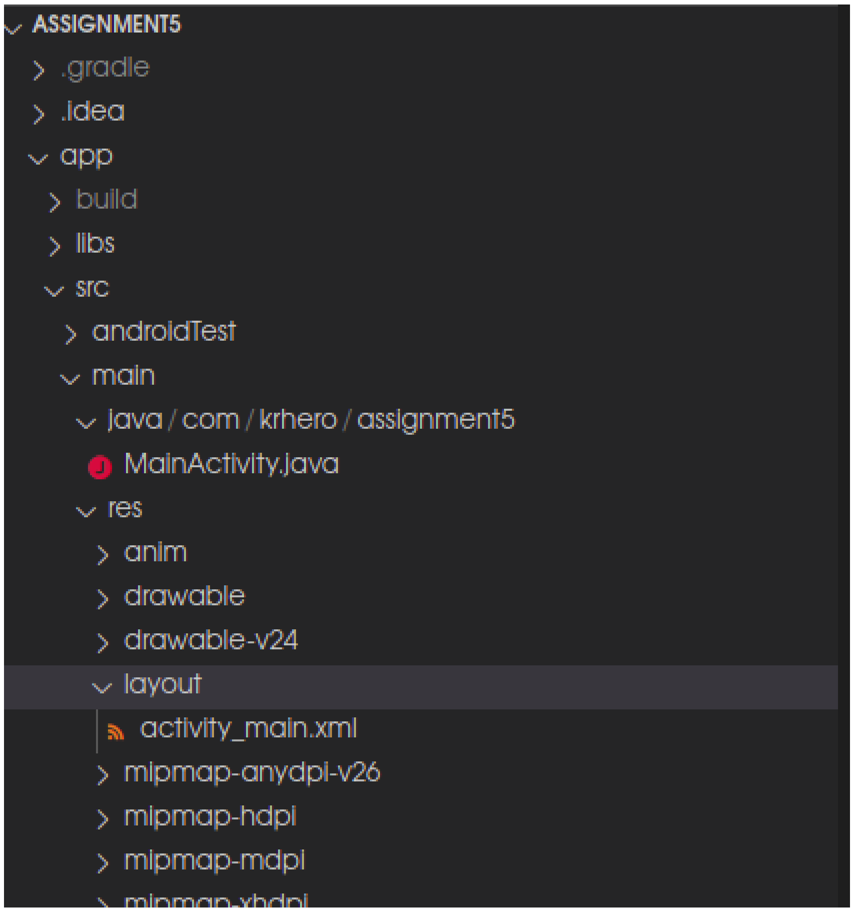
Software Tools 4
Assignment 5

Krunal Rank
U18CO081

Create an android application to make a simple calculator, which perform Addition, Subtraction, Multiplication, and Division.

Answer:

Directory Structure:



```
ASSIGNMENT5
  > .gradle
  > .idea
  v app
    > build
    > libs
    v src
      > androidTest
      v main
        v java / com / krhero / assignment5
            MainActivity.java
        v res
          > anim
          > drawable
          > drawable-v24
          v layout
              activity_main.xml
          > mipmap-anydpi-v26
          > mipmap-hdpi
          > mipmap-mdpi
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

```xml
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentBottom="true"
        android:layout_marginTop="0dp"
        android:fillViewport="true">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <TextView
                android:id="@+id/textView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentTop="true"
                android:layout_marginStart="10dp"
                android:layout_marginLeft="10dp"
                android:layout_marginTop="10dp"
                android:layout_marginEnd="10dp"
                android:layout_marginRight="10dp"
                android:fontFamily="sans-serif-light"
                android:gravity="bottom|right"
                android:lines="4"
                android:text="@string/String0"
                android:textSize="36sp" />

            <TableLayout
                android:id="@+id/numPad"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/textView"
                android:layout_marginStart="0dp"
                android:layout_marginLeft="0dp"
                android:layout_marginTop="100dp"
                android:layout_marginEnd="0dp"
                android:layout_marginRight="0dp">

                <TableRow
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
            android:gravity="right">

            <com.google.android.material.button.MaterialButton
                android:id="@+id/buttonDel"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/StringDel"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:textSize="18sp"
                app:backgroundTint="@android:color/holo_red_light"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/buttonClear"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/StringClear"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:textSize="18sp"
                app:backgroundTint="@android:color/holo_red_light"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

        </TableRow>

        <TableRow
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:gravity="center_horizontal"
            android:soundEffectsEnabled="false">

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```xml
            android:layout_margin="0dp"
            android:layout_weight="1"
            android:insetTop="0dp"
            android:insetBottom="0dp"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            android:text="@string/String1"
            android:textSize="18sp"
            app:cornerRadius="0dp"
            app:elevation="0dp" />

        <com.google.android.material.button.MaterialButton
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="0dp"
            android:layout_weight="1"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            android:insetTop="0dp"
            android:insetBottom="0dp"
            android:text="@string/String2"
            android:textSize="18sp"
            app:cornerRadius="0dp"
            app:elevation="0dp" />

        <com.google.android.material.button.MaterialButton
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="0dp"
            android:layout_weight="1"
            android:insetTop="0dp"
            android:insetBottom="0dp"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            android:text="@string/String3"
            android:textSize="18sp"
            app:cornerRadius="0dp"
            app:elevation="0dp" />

        <com.google.android.material.button.MaterialButton
            android:id="@+id/buttonAdd"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="0dp"
            android:layout_weight="1"
```

```
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/StringAdd"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:textSize="18sp"
                app:backgroundTint="@color/teal_700"
                app:cornerRadius="0dp"
                app:elevation="0dp" />
        </TableRow>

        <TableRow
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:gravity="center_horizontal">

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:insetBottom="0dp"
                android:text="@string/String4"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:text="@string/String5"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
```

```
                android:id="@+id/button6"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:text="@string/String6"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/buttonSub"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/StringSub"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:textSize="18sp"
                app:backgroundTint="@color/teal_700"
                app:cornerRadius="0dp"
                app:elevation="0dp" />
        </TableRow>

        <TableRow
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:gravity="center_horizontal">

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:insetBottom="0dp"
                android:text="@string/String7"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button8"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/String8"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button9"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:insetBottom="0dp"
                android:text="@string/String9"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/buttonMul"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/StringMul"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:textSize="18sp"
                app:backgroundTint="@color/teal_700"
```

```xml
                app:cornerRadius="0dp"
                app:elevation="0dp" />
        </TableRow>

        <TableRow
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:gravity="center_horizontal">

            <com.google.android.material.button.MaterialButton
                android:id="@+id/buttonDot"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:layout_weight="1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/StringDot"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/button0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                android:insetTop="0dp"
                android:insetBottom="0dp"
                android:text="@string/String0"
                android:textSize="18sp"
                app:cornerRadius="0dp"
                app:elevation="0dp" />

            <com.google.android.material.button.MaterialButton
                android:id="@+id/buttonEqual"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:layout_weight="1"
                android:insetTop="0dp"
```

```xml
                    android:insetBottom="0dp"
                    android:text="@string/StringEqual"
                    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                    android:textSize="18sp"
                    app:backgroundTint="@android:color/holo_green_light"
                    app:cornerRadius="0dp"
                    app:elevation="0dp" />

                <com.google.android.material.button.MaterialButton
                    android:id="@+id/buttonDiv"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_margin="0dp"
                    android:layout_weight="1"
                    android:insetTop="0dp"
                    android:insetBottom="0dp"
                    android:text="@string/StringDiv"
                    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                    android:textSize="18sp"
                    app:cornerRadius="0dp"
                    app:elevation="0dp"
                    app:backgroundTint="@color/teal_700" />
            </TableRow>

              </TableLayout>

       </RelativeLayout>
   </ScrollView>
</RelativeLayout>
```

strings.xml

```xml
<resources>
   <string name="app_name">CalC</string>
   <string name="String0">0</string>
   <string name="String1">1</string>
   <string name="String2">2</string>
   <string name="String3">3</string>
   <string name="String4">4</string>
   <string name="String5">5</string>
   <string name="String6">6</string>
   <string name="String7">7</string>
   <string name="String8">8</string>
   <string name="String9">9</string>
   <string name="StringDel">DEL</string>
   <string name="StringClear">CLR</string>
   <string name="StringEqual">=</string>
   <string name="StringAdd">+</string>
   <string name="StringSub">-</string>
   <string name="StringMul">*</string>
   <string name="StringDiv">/</string>
   <string name="StringDot">.</string>
   </resources>
```

MainActivity.java:

```java
package com.krhero.assignment5;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.TextView;

import com.google.android.material.button.MaterialButton;

import java.util.ArrayList;
import java.util.Stack;

public class MainActivity extends AppCompatActivity {

   private static class Symbol {
       int balancer;
       String id;
       double val = 0.0;
       String print = "";
       double multiplier = 1;

       public Symbol(String i, int b, double v, String p, double m) {
           this.id = i;
           this.balancer = b;
           this.val = v;
           this.print = p;
           this.multiplier = m;
       }

       public void debug() {
           System.out.println("DEBUG " + this.id + " " + this.balancer + " " +
this.val + " " + this.print + " "
                   + this.multiplier);
       }
   }

   private static class Literal {
       double val;
       String id, op;

       public Literal(double v, String i, String o) {
           this.val = v;
           this.id = i;
           this.op = o;
       }

       public void debug() {
           System.out.println("DEBUG " + this.id + " " + this.val + " " + this.op);
       }

   }

   private ArrayList<Symbol> stack;
   private ArrayList<MaterialButton> numButtons, opButtons;
   private MaterialButton add, sub, mul, div, equal, dot, clr, del;
   private TextView textView;
   private String ans;
   private Animation fadein;

   private void renderInfo() {
       ans = "";
       for (Symbol sym : stack) {
           ans += sym.print;
       }
       ;
       if (ans.length() == 0)
           ans = "0";
       textView.setText(ans);
       return;
   }

   private boolean isNum(String id) {
       return id.equals("1") || id.equals("2") || id.equals("3") || id.equals("4") ||
id.equals("5") || id.equals("6")
               || id.equals("7") || id.equals("8") || id.equals("9") ||
id.equals("0");
   }

   private boolean isBinaryOp(String id) {
       return id.equals("-") || id.equals("+") || id.equals("/") || id.equals("*") ||
id.equals("^") || id.equals("P")
```

```java
                || id.equals("C");
    }

    private boolean isTrigonometryOp(String id) {
        return id.equals("tan") || id.equals("sin") || id.equals("cos") ||
id.equals("atan") || id.equals("acos")
                || id.equals("asin") || id.equals("sinh") || id.equals("cosh") ||
id.equals("tanh");
    }

    private boolean isLogOp(String id) {
        return id.equals("log") || id.equals("log10");
    }

    private boolean isDot(String id) {
        return id.equals(".");
    }

    private boolean isFactorialOp(String id) {
        return id.equals("!");
    }

    private boolean isOpenBracket(String id) {
        return id.equals("(");
    }

    private boolean isCloseBracket(String id) {
        return id.equals(")");
    }

    private int getPrecedence(Literal l) {
        String id = l.op;
        String type = l.id;
        if (id.equals("(") || id.equals(")"))
            return -1;
        if (id.equals("+") || (id.equals("-") && type.equals("binaryOp")))
            return 0;
        if (id.equals("*"))
            return 1;
        if (id.equals("P") || id.equals("C"))
            return 2;
        if (id.equals("/"))
            return 3;
        if (id.equals("^"))
```

```
            return 4;
        if (isTrigonometryOp(id) || isLogOp(id))
            return 5;
        if (id.equals("-") && type.equals("unaryOp"))
            return 6;
        if (isFactorialOp(id))
            return 7;
        return 8;
    }

    private boolean isDouble(double val) {
        return Math.round(val) != val;
    }

    private double gamma(double z) {
        double g = 7;
        double[] C = {0.99999999999980993, 676.5203681218851, -1259.1392167224028,
771.32342877765313,
                -176.61502916214059, 12.507343278686905, -0.13857109526572012,
9.9843695780195716e-6,
                1.5056327351493116e-7};

        if (z < 0.5)
            return Math.PI / (Math.sin(Math.PI * z) * gamma(1 - z));
        else {
            z -= 1;

            double x = C[0];
            for (int i = 1; i < g + 2; i++)
                x += C[i] / (z + i);

            double t = z + g + 0.5;
            return Math.sqrt(2 * Math.PI) * Math.pow(t, (z + 0.5)) * Math.exp(-t) * x;
        }
    }

    private double factorial(double x) {
        return gamma(x + 1);
    }

    private void CLR() {
        stack.clear();
        renderInfo();
    }
```

```java
private void DEL() {
    if (stack.isEmpty()) {
        return;
    }
    Symbol top = stack.get(stack.size() - 1);
    stack.remove(stack.size() - 1);
    if (top.id == "(") {
        if (stack.isEmpty()) {
            renderInfo();
            return;
        }
        top = stack.get(stack.size() - 1);
        if (isTrigonometryOp(top.id) || isLogOp(top.id)) {
            stack.remove(stack.size() - 1);
        }

    }
    renderInfo();
}

private void ANS() {
    if (stack.isEmpty()) {
        renderInfo();
        return;
    }
    Symbol top = stack.get(stack.size() - 1);
    int extraBrackets = top.balancer;
    while (extraBrackets > 0) {
        stack.add(new Symbol(")", top.balancer - 1, -1, ")", -1));
        extraBrackets--;
    }
    ArrayList<Literal> arr = new ArrayList<Literal>();
    for (int i = 0; i < stack.size(); i++) {
        Symbol t = stack.get(i);
        if (isNum(t.id) || isDot(t.id)) {
            int j = i;
            double val = 0;
            while (isNum(t.id) || isDot(t.id)) {
                val = t.val;
                j++;
                if (j == stack.size())
                    break;
                t = stack.get(j);
                }

                arr.add(new Literal(val, "number", "X"));
                i = j - 1;
            } else if (isOpenBracket(t.id)) {
                arr.add(new Literal(-1, "openBracket", "X"));
            } else if (isCloseBracket(t.id)) {
                arr.add(new Literal(-1, "closeBracket", "X"));
            } else if (isBinaryOp(t.id)) {
                if (t.id.equals("-")) {
                    if (arr.size() == 0) {
                        arr.add(new Literal(-1, "unaryOp", t.id));
                    } else {
                        Literal l = arr.get(arr.size() - 1);
                        if (l.id.equals("binaryOp") || l.id.equals("openBracket")) {
                            arr.add(new Literal(-1, "unaryOp", t.id));
                        } else {
                            arr.add(new Literal(-1, "binaryOp", t.id));
                        }
                    }
                } else {
                    arr.add(new Literal(-1, "binaryOp", t.id));
                }
            } else if (isFactorialOp(t.id) || isTrigonometryOp(t.id) || isLogOp(t.id))
{
                arr.add(new Literal(-1, "unaryOp", t.id));
            }
        }
        try {
            ArrayList<Literal> postfix = new ArrayList<Literal>();
            Stack<Literal> st = new Stack<Literal>();
            for (int i = 0; i < arr.size(); i++) {
                Literal l = arr.get(i);
                if (l.id.equals("number"))
                    postfix.add(l);
                else if (l.id.equals("binaryOp") || l.id.equals("unaryOp")) {
                    if (st.isEmpty()) {
                        st.push(l);
                    } else {
                        while (!st.isEmpty() && getPrecedence(st.peek()) >=
getPrecedence(l)) {
                            if (st.peek().id.equals("openBracket")) {
                                break;
                            }
```

```java
                        postfix.add(st.peek());
                        st.pop();
                    }
                    st.push(l);
                }
            } else if (l.id.equals("openBracket")) {
                st.push(l);
            } else if (l.id.equals("closeBracket")) {
                while (!st.isEmpty() && !st.peek().id.equals("openBracket")) {
                    postfix.add(st.peek());
                    st.pop();
                }
                st.pop();
            }
        }
        while (!st.empty()) {
            postfix.add(st.peek());
            st.pop();
        }
        Stack<Double> finalStack = new Stack<Double>();
        for (int i = 0; i < postfix.size(); i++) {
            Literal l = postfix.get(i);
            if (l.id.equals("number")) {
                finalStack.push(l.val);
                continue;
            }
            double op1, op2, val;

            switch (l.op) {
                case "+":
                    op1 = finalStack.peek();
                    finalStack.pop();
                    if (finalStack.size() == 0) {
                        finalStack.push(op1);
                        break;
                    }
                    op2 = finalStack.peek();
                    finalStack.pop();
                    val = op2 + op1;
                    finalStack.push(val);
                    break;
                case "-":
                    if (l.id.equals("binaryOp")) {
                        op1 = finalStack.peek();
```

```
                            finalStack.pop();
                            op2 = finalStack.peek();
                            finalStack.pop();
                            val = op2 - op1;
                            finalStack.push(val);
                        } else {
                            op1 = finalStack.peek();
                            finalStack.pop();
                            val = -op1;
                            finalStack.push(val);
                        }
                        break;
                    case "*":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        op2 = finalStack.peek();
                        finalStack.pop();
                        val = op2 * op1;
                        finalStack.push(val);
                        break;
                    case "/":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        op2 = finalStack.peek();
                        finalStack.pop();
                        val = op2 / op1;
                        finalStack.push(val);
                        break;
                    case "^":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        op2 = finalStack.peek();
                        finalStack.pop();
                        val = Math.pow(op2, op1);
                        finalStack.push(val);
                        break;
                    case "P":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        op2 = finalStack.peek();
                        finalStack.pop();
                        val = ((op2 < 0 ? -1 : 1) * factorial(op2)) / ((op2 < op1 ? -1
: 1) * factorial(op2 - op1));
                        finalStack.push(val);
                        break;
                    case "C":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        op2 = finalStack.peek();
                        finalStack.pop();
                        val = ((op2 < 0 ? -1 : 1) * factorial(op2)) / (((op2 < op1 ? -1
 : 1) * factorial(op2 - op1))
                                * ((op1 < 0 ? -1 : 1) * factorial(op1)));
                        finalStack.push(val);
                        break;
                    case "!":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        val = (op1 < 0 ? -1 : 1) * factorial(op1);
                        finalStack.push(val);
                        break;
                    case "sin":
                        op1 = finalStack.peek();
                        System.out.println(op1);
                        finalStack.pop();
                        finalStack.push(Math.sin(op1));
                        break;
                    case "cos":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        finalStack.push(Math.cos(op1));
                        break;
                    case "tan":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        finalStack.push(Math.tan(op1));
                        break;
                    case "asin":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        finalStack.push(Math.asin(op1));
                        break;
                    case "acos":
                        op1 = finalStack.peek();
                        finalStack.pop();
                        finalStack.push(Math.acos(op1));
                        break;
                    case "atan":
```

```java
                    op1 = finalStack.peek();
                    finalStack.pop();
                    finalStack.push(Math.atan(op1));
                    break;
                case "sinh":
                    op1 = finalStack.peek();
                    finalStack.pop();
                    finalStack.push(Math.sinh(op1));
                    break;
                case "cosh":
                    op1 = finalStack.peek();
                    finalStack.pop();
                    finalStack.push(Math.cosh(op1));
                    break;
                case "tanh":
                    op1 = finalStack.peek();
                    finalStack.pop();
                    finalStack.push(Math.tanh(op1));
                    break;
                case "log":
                    op1 = finalStack.peek();
                    finalStack.pop();
                    finalStack.push(Math.log(op1));
                    break;
                case "log10":
                    op1 = finalStack.peek();
                    finalStack.pop();
                    finalStack.push(Math.log10(op1));
                    break;
                default:
            }
        }
        stack.clear();
        double finalAns = 0;
        while (!finalStack.empty()) {
            finalAns += finalStack.peek();
            finalStack.pop();
        }
        if (finalAns == 0)
            return;
        String s = String.format("%.4f", finalAns);
        for (int i = 0; i < s.length(); i++) {
            String t = "" + s.charAt(i);
            if (isNum(t))
```

```java
                    pressDigit(Integer.parseInt(t));
                if (isDot(t))
                    pressDot(".");
                if (isBinaryOp(t))
                    pressBinaryOp(t);
            }

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    private void EXIT() {
        System.exit(0);
    }

    private void pressDigit(int n) {
        if (n == 0) {
            if (stack.isEmpty())
                return;
            Symbol top = stack.get(stack.size() - 1);
            if (top.val == 0) {
                if (top.multiplier >= 0 && top.multiplier <= 1) {
                    stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n,
top.multiplier * 0.1));
                    renderInfo();
                    return;
                } else {
                    return;
                }
            }
        }
        if (stack.isEmpty()) {
            stack.add(new Symbol("" + n, 0, (double) n, "" + n, 1));
            renderInfo();
            return;
        }
        Symbol top = stack.get(stack.size() - 1);
        if (isNum(top.id)) {
            if (top.val == 0 && !(top.multiplier > 0 && top.multiplier < 1))
                stack.remove(stack.size() - 1);
            stack.add(new Symbol("" + n, top.balancer,
                    top.multiplier >= 0 && top.multiplier < 1 ? top.val + n *
top.multiplier : top.val * 10 + n, "" + n,
                    top.multiplier >= 0 && top.multiplier < 1 ? top.multiplier / 10.0 :
 1));
        } else if (isBinaryOp(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n, 1));
        } else if (isTrigonometryOp(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n, 1));
        } else if (isLogOp(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n, 1));
        } else if (isDot(top.id)) {
            stack.add(new Symbol("" + n, top.balancer,
                    top.multiplier >= 0 && top.multiplier < 1 ? top.val + n *
 top.multiplier : top.val * 10 + n, "" + n,
                    0.01));
        } else if (isFactorialOp(top.id)) {
            stack.add(new Symbol("*", top.balancer, -1, "*", 1));
            stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n, 1));
        } else if (isOpenBracket(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n, 1));
        } else if (isCloseBracket(top.id)) {
            stack.add(new Symbol("*", top.balancer, -1, "*", 1));
            stack.add(new Symbol("" + n, top.balancer, (double) n, "" + n, 1));
        }
        renderInfo();
    }

    private void pressBinaryOp(String n) {
        if (stack.isEmpty()) {
            if (n.equals("-")) {
                stack.add(new Symbol("" + n, 0, -1, "" + n, -1));
                renderInfo();
            }
            return;
        }
        Symbol top = stack.get(stack.size() - 1);
        if (isNum(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
        } else if (isBinaryOp(top.id)) {
            stack.remove(stack.size() - 1);
            stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
        } else if (isTrigonometryOp(top.id)) {
            if (n.equals("-")) {
                stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
            }
        } else if (isLogOp(top.id)) {
```

```java
            if (n.equals("-")) {
                stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
            }
        } else if (isDot(top.id)) {
            stack.remove(stack.size() - 1);
            stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
        } else if (isFactorialOp(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
        } else if (isOpenBracket(top.id)) {
            if (n.equals("-")) {
                stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
            }
        } else if (isCloseBracket(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, -1, "" + n, -1));
        }
        renderInfo();
    }

    private void pressDot(String n) {
        if (stack.isEmpty()) {
            stack.add(new Symbol("0", 0, 0, "0", 1));
            pressDot(".");
            renderInfo();
            return;
        }
        Symbol top = stack.get(stack.size() - 1);
        if (top.multiplier > 0 && top.multiplier < 1) return;
        if (isNum(top.id)) {
            stack.add(new Symbol("" + n, top.balancer, top.val, "" + n, 0.1));
        } else if (isBinaryOp(top.id)) {
            pressDigit(0);
            pressDot(".");
        } else if (isTrigonometryOp(top.id)) {
            pressDigit(0);
            pressDot(".");
        } else if (isLogOp(top.id)) {
            pressDigit(0);
            pressDot(".");
        } else if (isDot(top.id)) {
        } else if (isFactorialOp(top.id)) {
            pressBinaryOp("*");
            pressDigit(0);
            pressDot(".");
        } else if (isOpenBracket(top.id)) {
            pressDigit(0);
            pressDot(".");
        } else if (isCloseBracket(top.id)) {
            pressBinaryOp("*");
            pressDigit(0);
            pressDot(".");
        }
        renderInfo();

    }

    private void pressInverseOp(String n) {
        if (stack.isEmpty())
            return;
        Symbol top = stack.get(stack.size() - 1);
        if (top.val == 0)
            return;
        ArrayList<Symbol> newStack = new ArrayList<Symbol>();
        newStack.add(new Symbol("1", 0, 1, "1", 1));
        newStack.add(new Symbol("/", 0, -1, "/", -1));
        newStack.add(new Symbol("(", 1, -1, "(", -1));
        for (int i = 0; i < stack.size(); i++) {
            Symbol sym = stack.get(i);
            sym.balancer += 1;
            newStack.add(sym);
        }
        stack.clear();
        for (int i = 0; i < newStack.size(); i++)
            stack.add(newStack.get(i));
        newStack.clear();
        renderInfo();
        return;
    }

    private void pressOpenBracket() {
        if (stack.isEmpty()) {
            stack.add(new Symbol("(", 1, -1, "(", -1));
            renderInfo();
            return;
        }
        Symbol top = stack.get(stack.size() - 1);
        if (isNum(top.id)) {
            pressBinaryOp("*");
            pressOpenBracket();
```

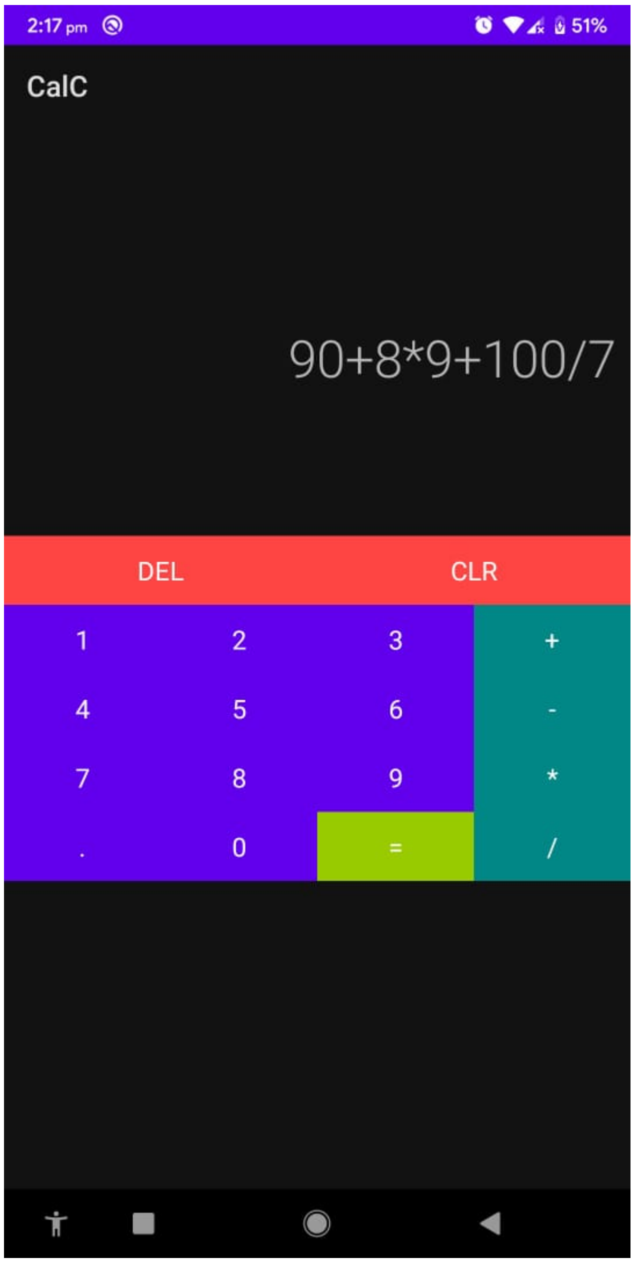
```java
    } else if (isBinaryOp(top.id)) {
        stack.add(new Symbol("(", top.balancer + 1, -1, "(", -1));
    } else if (isTrigonometryOp(top.id)) {
        stack.add(new Symbol("(", top.balancer + 1, -1, "(", -1));
    } else if (isLogOp(top.id)) {
        stack.add(new Symbol("(", top.balancer + 1, -1, "(", -1));
    } else if (isDot(top.id)) {
        stack.remove(stack.size() - 1);
        pressBinaryOp("*");
        pressOpenBracket();
    } else if (isFactorialOp(top.id)) {
        pressBinaryOp("*");
        pressOpenBracket();
    } else if (isOpenBracket(top.id)) {
        stack.add(new Symbol("(", top.balancer + 1, -1, "(", -1));
    } else if (isCloseBracket(top.id)) {
        pressBinaryOp("*");
        pressOpenBracket();
    }
    renderInfo();
}

private void pressCloseBracket() {
    if (stack.isEmpty()) {
        return;
    }

    Symbol top = stack.get(stack.size() - 1);
    if (top.val == 0)
        return;
    if (top.balancer <= 0)
        return;
    if (isNum(top.id)) {
        stack.add(new Symbol(")", top.balancer - 1, -1, ")", -1));
    } else if (isBinaryOp(top.id)) {
    } else if (isTrigonometryOp(top.id)) {
    } else if (isLogOp(top.id)) {
    } else if (isDot(top.id)) {
        stack.remove(stack.size() - 1);
        stack.add(new Symbol(")", top.balancer - 1, -1, ")", -1));
    } else if (isFactorialOp(top.id)) {
        stack.add(new Symbol(")", top.balancer - 1, -1, ")", -1));
    } else if (isOpenBracket(top.id)) {
    } else if (isCloseBracket(top.id)) {
        stack.add(new Symbol(")", top.balancer - 1, -1, ")", -1));
    }
    renderInfo();
}

private void pressFactorial() {
    if (stack.isEmpty()) {
        return;
    }
    Symbol top = stack.get(stack.size() - 1);
    if (isNum(top.id)) {
        stack.add(new Symbol("!", top.balancer, -1, "!", -1));
    }
    renderInfo();
}

private void constantHelper(String n) {
    String s = "";
    if (n.equals("pi"))
        s = String.format("%.14f", Math.PI);
    if (n.equals("exp"))
        s = String.format("%.14f", Math.E);
    for (int i = 0; i < s.length(); i++) {
        if (isNum("" + s.charAt(i)))
            pressDigit(Integer.parseInt("" + s.charAt(i)));
        if (isDot("" + s.charAt(i)))
            pressDot(".");
        if (isBinaryOp("" + s.charAt(i)))
            pressBinaryOp("" + s.charAt(i));
    }
}

private void pressConstant(String n) {
    if (stack.isEmpty()) {
        constantHelper(n);
        return;
    }
    Symbol top = stack.get(stack.size() - 1);
    if (isNum(top.id)) {
        pressBinaryOp("*");
        pressConstant(n);
    } else if (isBinaryOp(top.id)) {
        constantHelper(n);
    } else if (isTrigonometryOp(top.id)) {
        constantHelper(n);
    } else if (isLogOp(top.id)) {
        constantHelper(n);
    } else if (isDot(top.id)) {
        stack.remove(stack.size() - 1);
        pressBinaryOp("*");
        constantHelper(n);
    } else if (isFactorialOp(top.id)) {
        pressBinaryOp("*");
        constantHelper(n);
    } else if (isOpenBracket(top.id)) {
        constantHelper(n);
    } else if (isCloseBracket(top.id)) {
        pressBinaryOp("*");
        constantHelper(n);
    }
}

private void pressTrigonometryOp(String t) {
    if (stack.isEmpty()) {
        stack.add(new Symbol(t, 0, -1, t, -1));
        pressOpenBracket();
        renderInfo();
        return;
    }
    Symbol top = stack.get(stack.size() - 1);
    if (isNum(top.id)) {
        pressBinaryOp("*");
        pressTrigonometryOp(t);
    } else if (isBinaryOp(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isTrigonometryOp(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isLogOp(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isDot(top.id)) {
        stack.remove(stack.size() - 1);
        pressBinaryOp("*");
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isFactorialOp(top.id)) {
        pressBinaryOp("*");
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isOpenBracket(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isCloseBracket(top.id)) {
        pressBinaryOp("*");
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    }
    renderInfo();
}

private void pressLogOp(String t) {
    if (stack.isEmpty()) {
        stack.add(new Symbol(t, 0, -1, t, -1));
        pressOpenBracket();
        renderInfo();
        return;
    }
    Symbol top = stack.get(stack.size() - 1);
    if (isNum(top.id)) {
        pressBinaryOp("*");
        pressLogOp(t);
    } else if (isBinaryOp(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isTrigonometryOp(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isLogOp(top.id)) {
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isDot(top.id)) {
        stack.remove(stack.size() - 1);
        pressBinaryOp("*");
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isFactorialOp(top.id)) {
        pressBinaryOp("*");
        stack.add(new Symbol(t, top.balancer, -1, t, -1));
        pressOpenBracket();
    } else if (isOpenBracket(top.id)) {
```

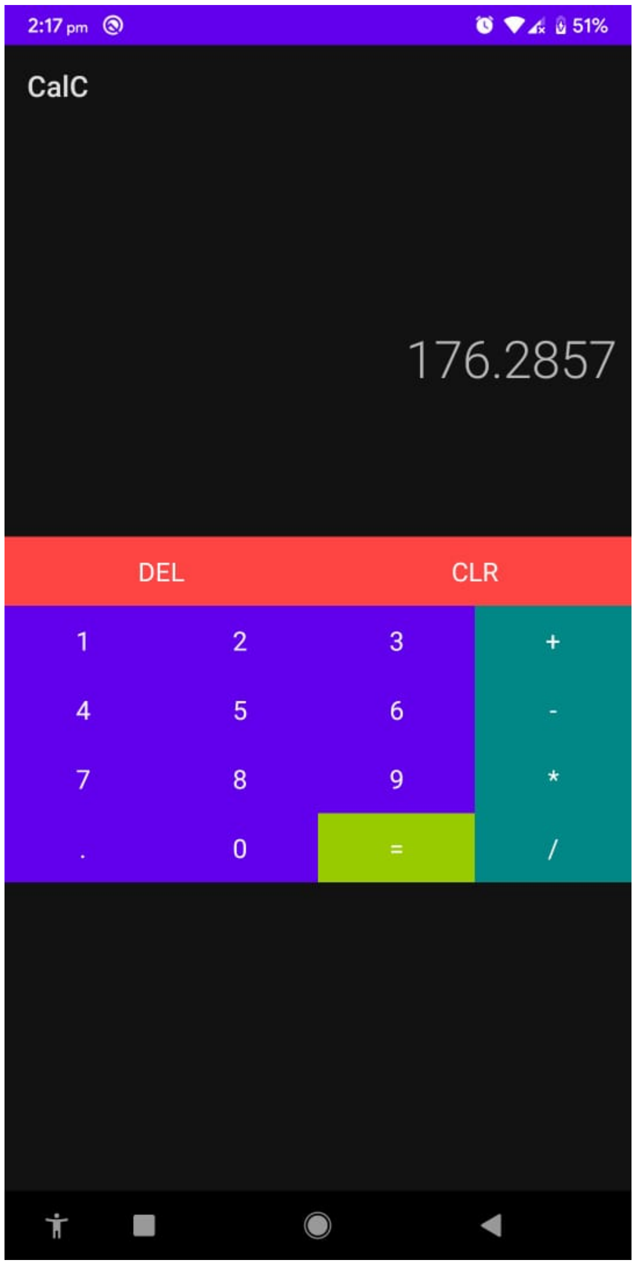
```java
            stack.add(new Symbol(t, top.balancer, -1, t, -1));
            pressOpenBracket();
        } else if (isCloseBracket(top.id)) {
            pressBinaryOp("*");
            stack.add(new Symbol(t, top.balancer, -1, t, -1));
            pressOpenBracket();
        }
        renderInfo();
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fadein = AnimationUtils.loadAnimation(this, R.anim.fadein);

        textView = (TextView) findViewById(R.id.textView);
        textView.setAnimation(fadein);

        add = (MaterialButton) findViewById(R.id.buttonAdd);
        sub = (MaterialButton) findViewById(R.id.buttonSub);
        mul = (MaterialButton) findViewById(R.id.buttonMul);
        div = (MaterialButton) findViewById(R.id.buttonDiv);
        equal = (MaterialButton) findViewById(R.id.buttonEqual);
        dot = (MaterialButton) findViewById(R.id.buttonDot);
        clr = (MaterialButton) findViewById(R.id.buttonClear);
        del = (MaterialButton) findViewById(R.id.buttonDel);


        numButtons = new ArrayList<MaterialButton>();
        stack = new ArrayList<Symbol>();

        numButtons.add((MaterialButton) findViewById(R.id.button0));
        numButtons.add((MaterialButton) findViewById(R.id.button1));
        numButtons.add((MaterialButton) findViewById(R.id.button2));
        numButtons.add((MaterialButton) findViewById(R.id.button3));
        numButtons.add((MaterialButton) findViewById(R.id.button4));
        numButtons.add((MaterialButton) findViewById(R.id.button5));
        numButtons.add((MaterialButton) findViewById(R.id.button6));
        numButtons.add((MaterialButton) findViewById(R.id.button7));
        numButtons.add((MaterialButton) findViewById(R.id.button8));
        numButtons.add((MaterialButton) findViewById(R.id.button9));
```

```java
for (MaterialButton numButton : numButtons) {
    numButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            int num = Integer.parseInt("" + numButton.getText());
            pressDigit(num);
        }
    });
}

equal.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ANS();
    }
});

div.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pressBinaryOp("/");
    }
});

mul.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pressBinaryOp("*");
    }
});
add.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pressBinaryOp("+");
    }
});
sub.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pressBinaryOp("-");
```

```
            }
        });
        dot.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pressDot(".");
            }
        });
        clr.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                CLR();
            }
        });
        del.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                DEL();
            }
        });

        del.setOnLongClickListener(new View.OnLongClickListener() {

            @Override
            public boolean onLongClick(View v) {
                CLR();
                return true;
            }

            ;
        });


    }
}
```

Screenshots:


2:17 pm
51%
CalC
90+8*9+100/7
DEL
CLR
1
2
3
+
4
5
6
-
7
8
9
*
.
0
=
/


2:17 pm
51%
CalC
176.2857
DEL
CLR
1
2
3
+
4
5
6
-
7
8
9
*
.
0
=
/