Control Flow Graph Testing

Software Testing

—

Name

Shivam Jain (MT2020002)

Vaibhav Tandon (MT2020150)

Rahul Vishnoi (MT2020076)

Overview

The aim of this project is to perform Control Flow Graph testing on a sample code that covers the edge and prime path for all the functions in the code.

Basic Code Description

We have used a calculator which calculates Body Mass Index for an individual and computes scientific calculations such as finding factorial, square root, power, finding division, and simple string operations also.

Testing Strategy Used

Control Flow Graph

It is the graphical representation of control flow or computation during the execution of programs or applications. It is mostly used in static analysis and compiler applications. We have used control flow graph testing to test our code.

Testing Tool Used

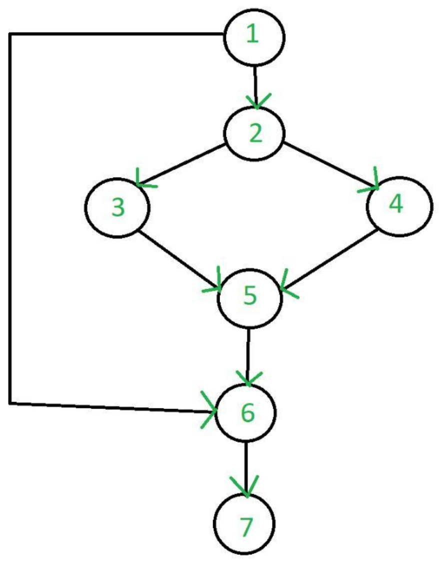
We have used JUnit to design our test cases. **JUnit** is a testing framework for Java programming language which is used for unique testing of code.

Test Case Description

We have designed our test cases in such a way that it covers every edge of a control flow graph and cover the prime path for a control flow graph.

CFG for a sample code:-

```
if  A = 10 then
  if B > C
     A = B
  else A = C
  endif
  endif
print A, B, C
```



Control Flow Graph

I. Edge Coverage

Edge coverage includes designing a test case in such a way that it covers every edge of a control flow graph.

Test Cases that cover the edge of the above control flow graph are as follows:-

1. [1, 2, 5, 6, 7]
2. [1, 4, 5, 6, 7]
3. [1, 6, 7]

II. Prime Path Coverage

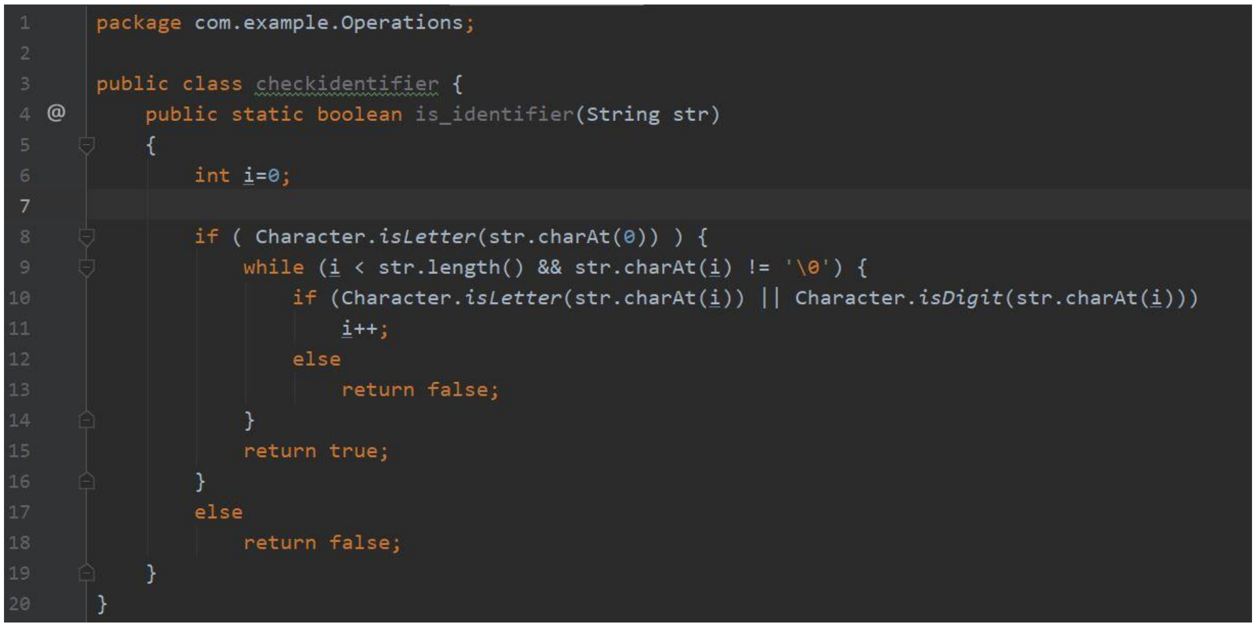
A prime path is basically *a* simple path and it does not appear as a sub-path of any other simple path.

Test Cases that cover the prime path of the above control flow graph are as follows:-

1. [1, 2, 5, 6, 7]
2. [1, 4, 5, 6, 7]
3. [1, 6, 7]

Control Flow Graph for the Code used

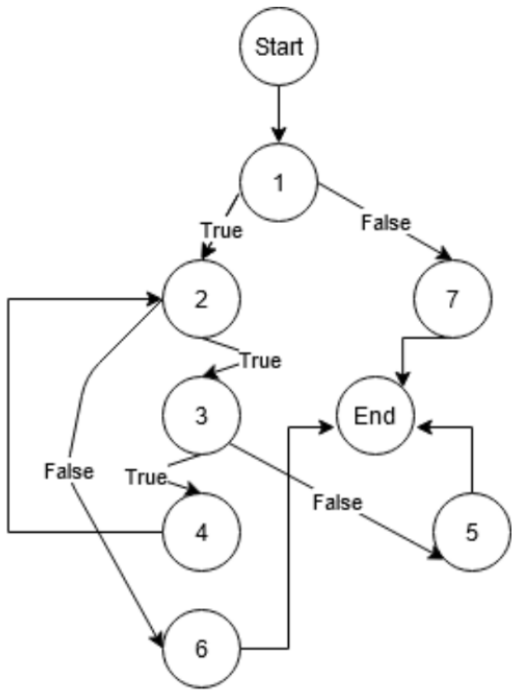
Check Identifier



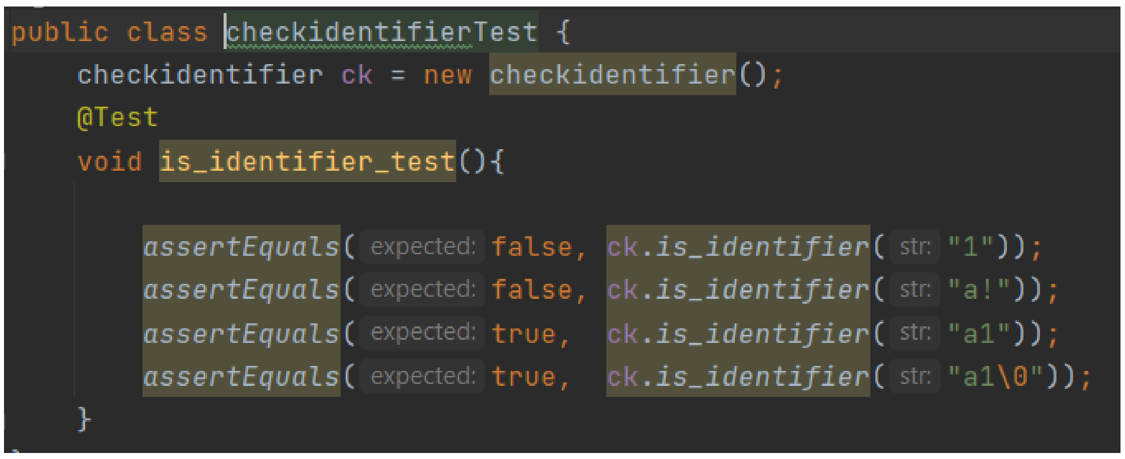
```
1   package com.example.Operations;
2
3   public class checkidentifier {
4 @     public static boolean is_identifier(String str)
5       {
6           int i=0;
7
8           if ( Character.isLetter(str.charAt(0)) ) {
9               while (i < str.length() && str.charAt(i) != '\0') {
10                  if (Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
11                      i++;
12                  else
13                      return false;
14              }
15              return true;
16          }
17          else
18              return false;
19      }
20  }
```

Block Number:-

Node / Block Number	Line Number
1	6,8
2	9
3	10
4	11
5	13
6	15
7	18



Test Cases:-



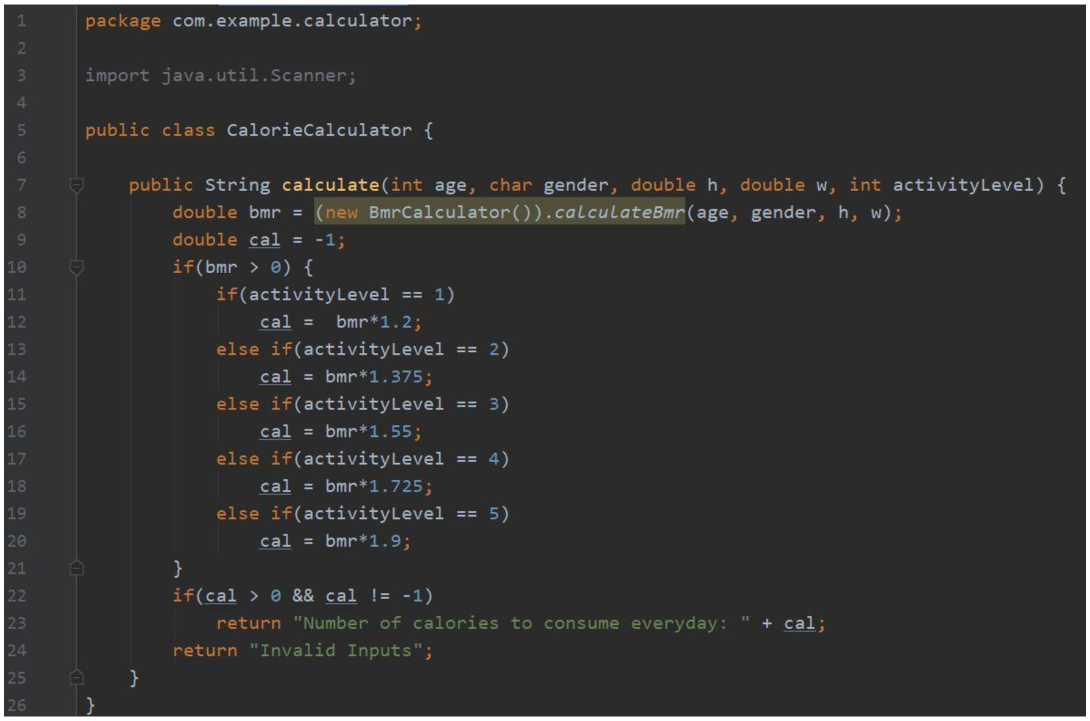
```
public class checkidentifierTest {
    checkidentifier ck = new checkidentifier();
    @Test
    void is_identifier_test(){

        assertEquals( expected: false, ck.is_identifier( str: "1"));
        assertEquals( expected: false, ck.is_identifier( str: "a!"));
        assertEquals( expected: true,  ck.is_identifier( str: "a1"));
        assertEquals( expected: true,  ck.is_identifier( str: "a1\0"));
    }
```

The above Test cases cove the following path:-

1. [1, 7, end]
2. [1, 2, 6, end]
3. [1, 2, 3, 4, 2, 3, 4, 6, end]
4. [1, 2, 3, 4, 2, 3, 4, 2, 3, 5, end]

These paths cover all the prime paths and edges.

Calorie Calculator

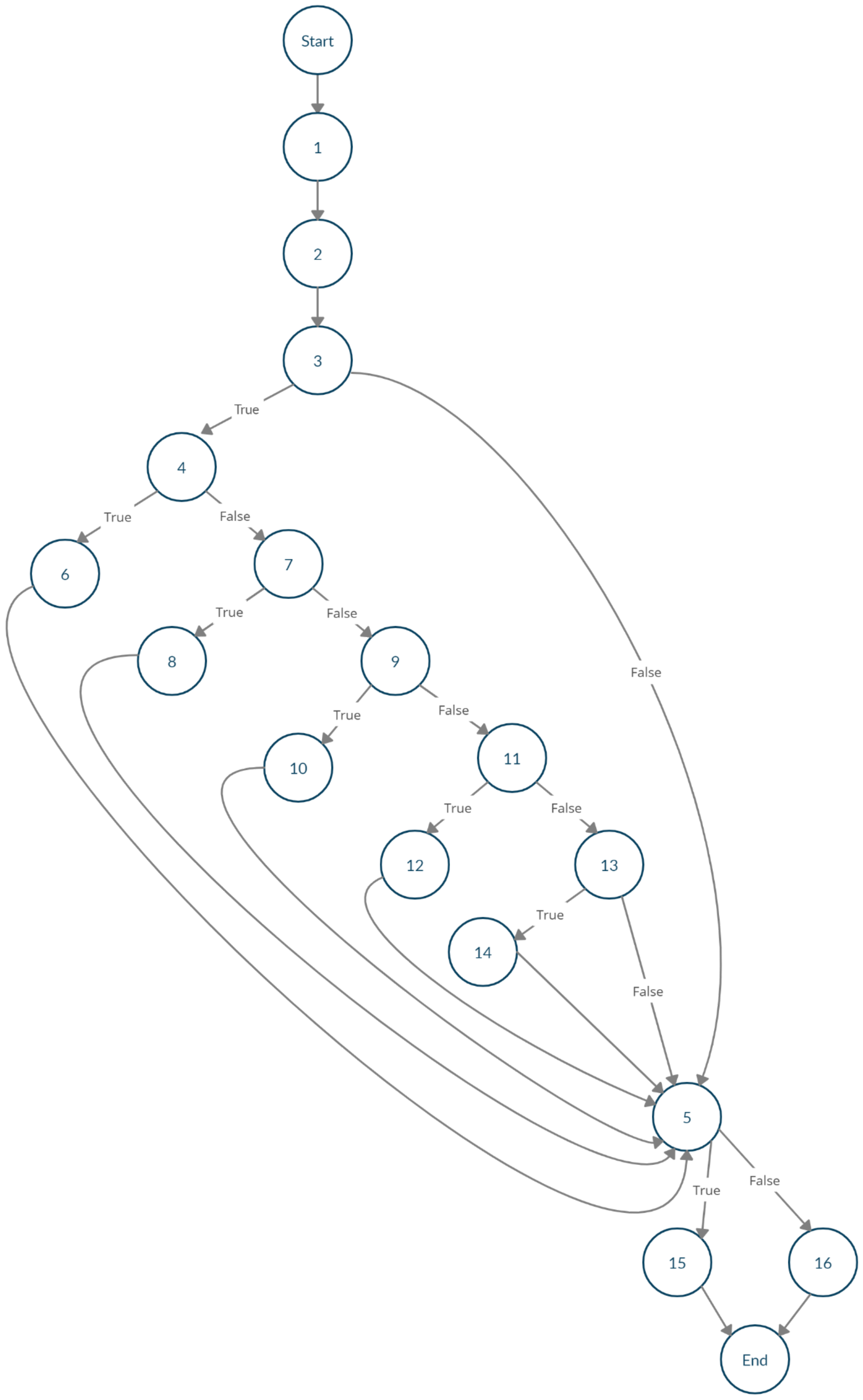


```
1  package com.example.calculator;
2
3  import java.util.Scanner;
4
5  public class CalorieCalculator {
6
7      public String calculate(int age, char gender, double h, double w, int activityLevel) {
8          double bmr = (new BmrCalculator()).calculateBmr(age, gender, h, w);
9          double cal = -1;
10         if(bmr > 0) {
11             if(activityLevel == 1)
12                 cal =  bmr*1.2;
13             else if(activityLevel == 2)
14                 cal = bmr*1.375;
15             else if(activityLevel == 3)
16                 cal = bmr*1.55;
17             else if(activityLevel == 4)
18                 cal = bmr*1.725;
19             else if(activityLevel == 5)
20                 cal = bmr*1.9;
21         }
22         if(cal > 0 && cal != -1)
23             return "Number of calories to consume everyday: " + cal;
24         return "Invalid Inputs";
25     }
26 }
```

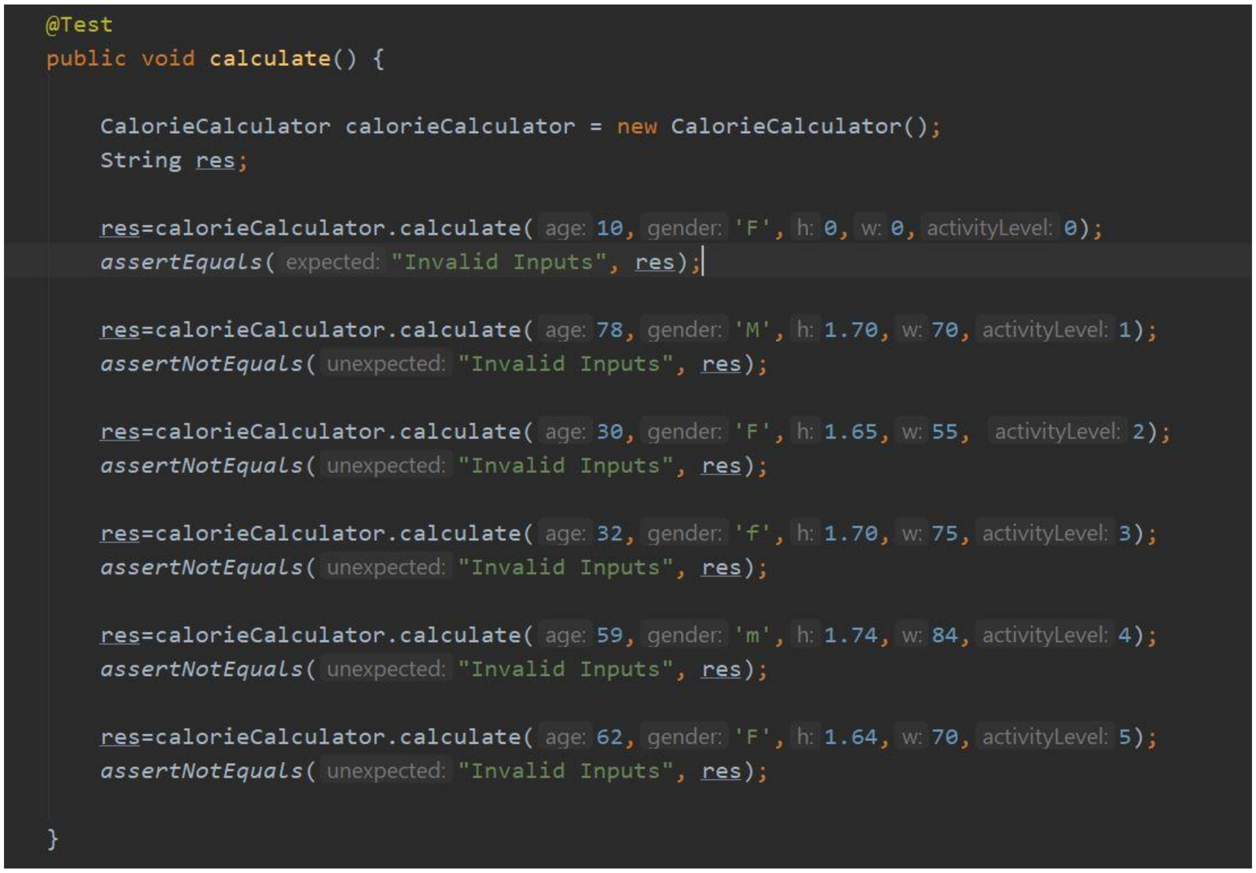
Block Number

Node/Block Number	Line Number (Refer Code)
1	8
2	9
3	10
4	11
5	22
6	12
7	13
8	14
9	15
10	16
11	17
12	18
13	19

14	20
15	23
16	24



Test Cases -

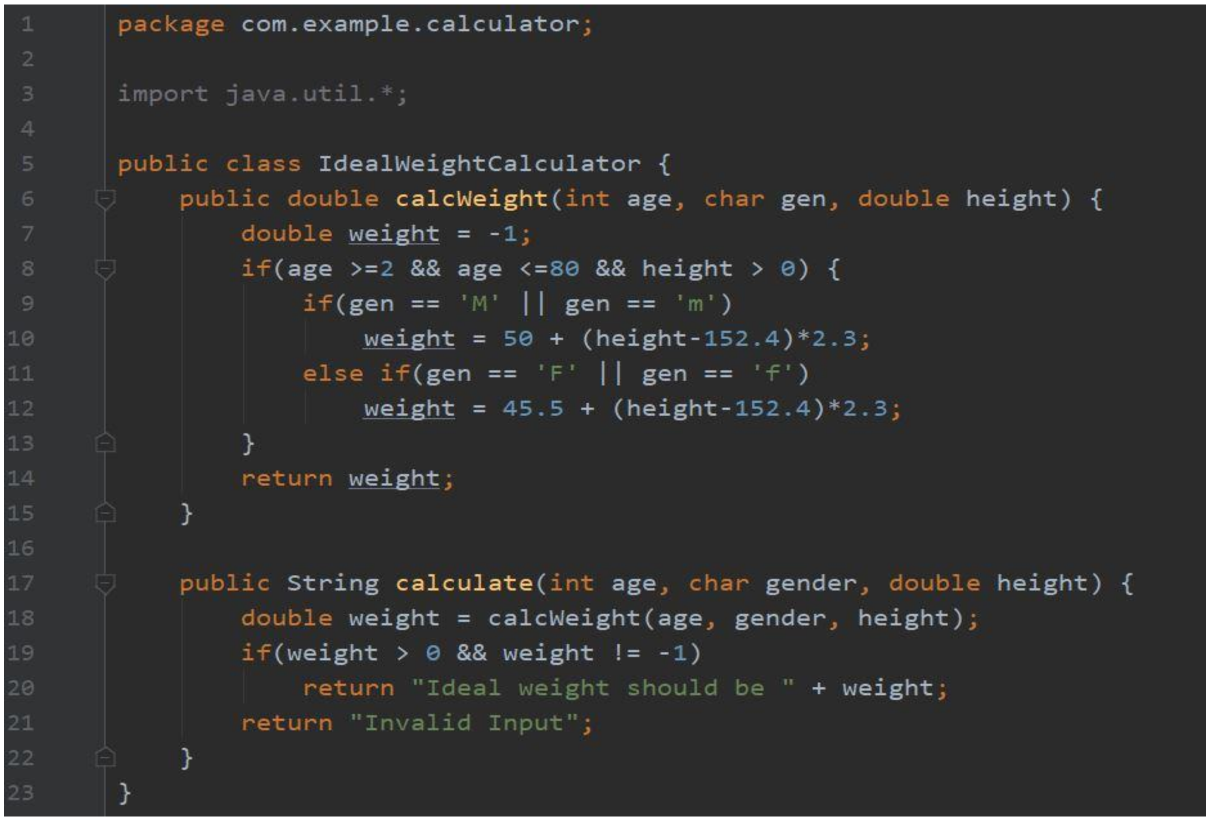


```java
@Test
public void calculate() {

    CalorieCalculator calorieCalculator = new CalorieCalculator();
    String res;

    res=calorieCalculator.calculate( age: 10, gender: 'F', h: 0, w: 0, activityLevel: 0);
    assertEquals( expected: "Invalid Inputs", res);

    res=calorieCalculator.calculate( age: 78, gender: 'M', h: 1.70, w: 70, activityLevel: 1);
    assertNotEquals( unexpected: "Invalid Inputs", res);

    res=calorieCalculator.calculate( age: 30, gender: 'F', h: 1.65, w: 55,  activityLevel: 2);
    assertNotEquals( unexpected: "Invalid Inputs", res);

    res=calorieCalculator.calculate( age: 32, gender: 'f', h: 1.70, w: 75, activityLevel: 3);
    assertNotEquals( unexpected: "Invalid Inputs", res);

    res=calorieCalculator.calculate( age: 59, gender: 'm', h: 1.74, w: 84, activityLevel: 4);
    assertNotEquals( unexpected: "Invalid Inputs", res);

    res=calorieCalculator.calculate( age: 62, gender: 'F', h: 1.64, w: 70, activityLevel: 5);
    assertNotEquals( unexpected: "Invalid Inputs", res);

}
```

The above Test cases cove the following path:-

1. [Start,1,2,3,5,16,End]
2. [Start,1,2,3,4,6,5,15,End]
3. [Start,1,2,3,4,7,8,5,15,End]
4. [Start,1,2,3,4,7,9,10,5,15,End]
5. [Start,1,2,3,4,7,9,11,12,5,15,End]
6. [Start,1,2,3,4,7,9,11,13,14,5,15,End]

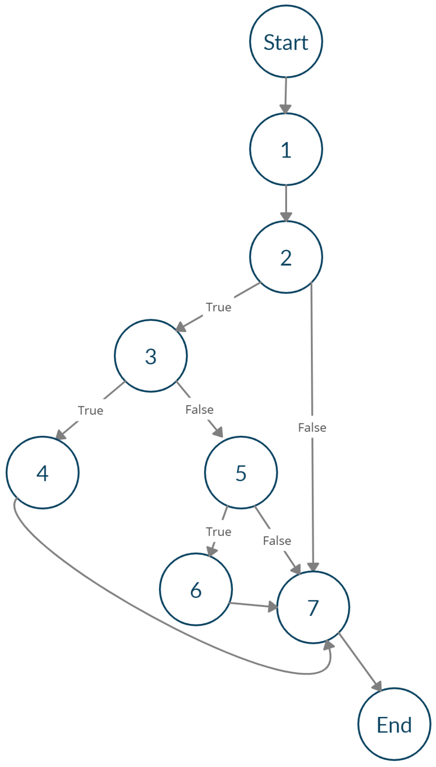
Ideal Weight Calculator



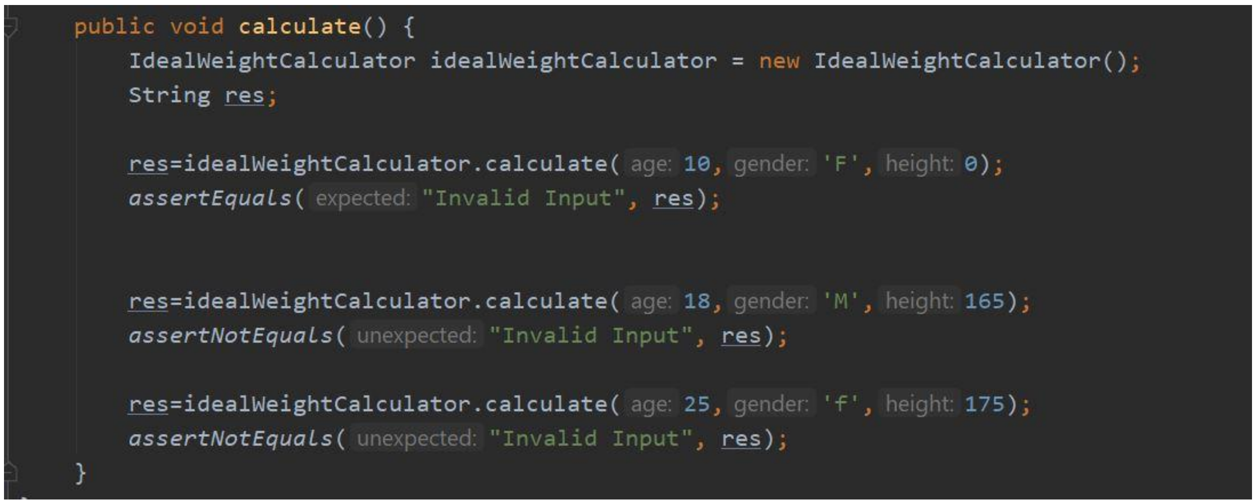
```
1   package com.example.calculator;
2
3   import java.util.*;
4
5   public class IdealWeightCalculator {
6       public double calcWeight(int age, char gen, double height) {
7           double weight = -1;
8           if(age >=2 && age <=80 && height > 0) {
9               if(gen == 'M' || gen == 'm')
10                  weight = 50 + (height-152.4)*2.3;
11              else if(gen == 'F' || gen == 'f')
12                  weight = 45.5 + (height-152.4)*2.3;
13          }
14          return weight;
15      }
16
17      public String calculate(int age, char gender, double height) {
18          double weight = calcWeight(age, gender, height);
19          if(weight > 0 && weight != -1)
20              return "Ideal weight should be " + weight;
21          return "Invalid Input";
22      }
23  }
```

Block Numbers :-

Node Number	Line Numbers (Refer Code)
1	7
2	8
3	9
4	10
5	11
6	12
7	14



Test Cases -



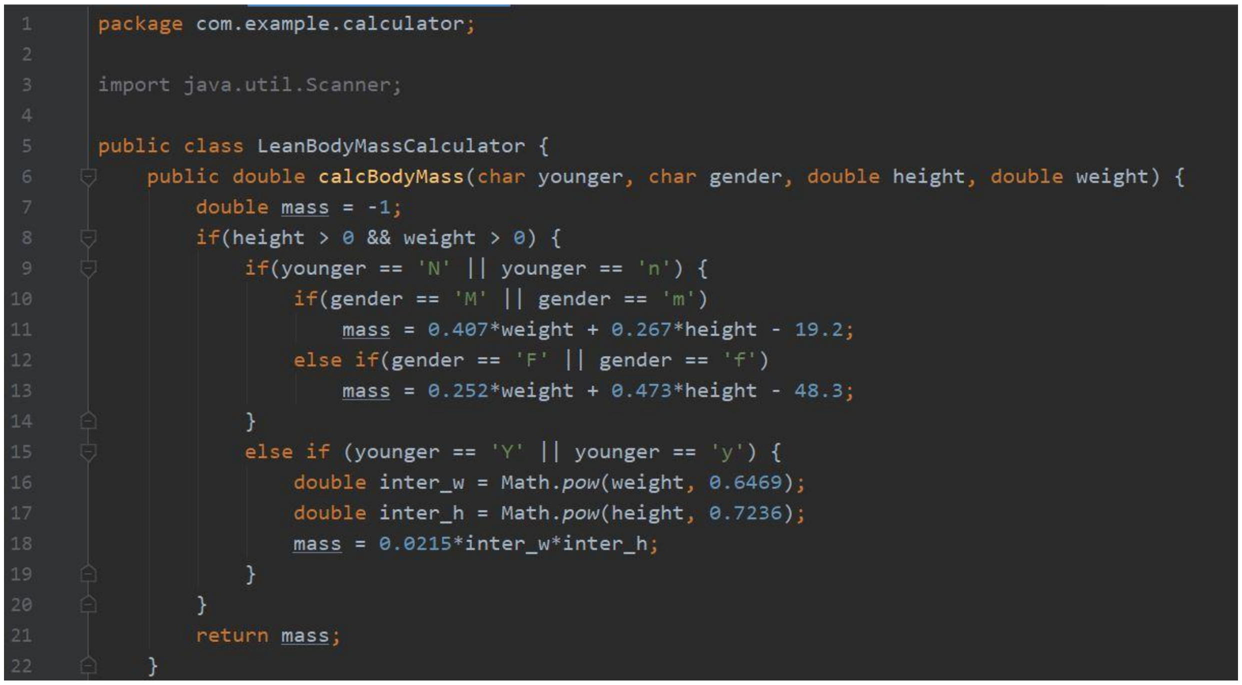
```
public void calculate() {
    IdealWeightCalculator idealWeightCalculator = new IdealWeightCalculator();
    String res;

    res=idealWeightCalculator.calculate( age: 10, gender: 'F', height: 0);
    assertEquals( expected: "Invalid Input", res);


    res=idealWeightCalculator.calculate( age: 18, gender: 'M', height: 165);
    assertNotEquals( unexpected: "Invalid Input", res);

    res=idealWeightCalculator.calculate( age: 25, gender: 'f', height: 175);
    assertNotEquals( unexpected: "Invalid Input", res);
}
```

The above Test cases cove the following path:-

1. [1, 2, 7, end]
2. [1, 2, 3, 4, 7, end]
3. [1, 2, 3, 5, 6, end]

These paths cover all the prime paths and edges.

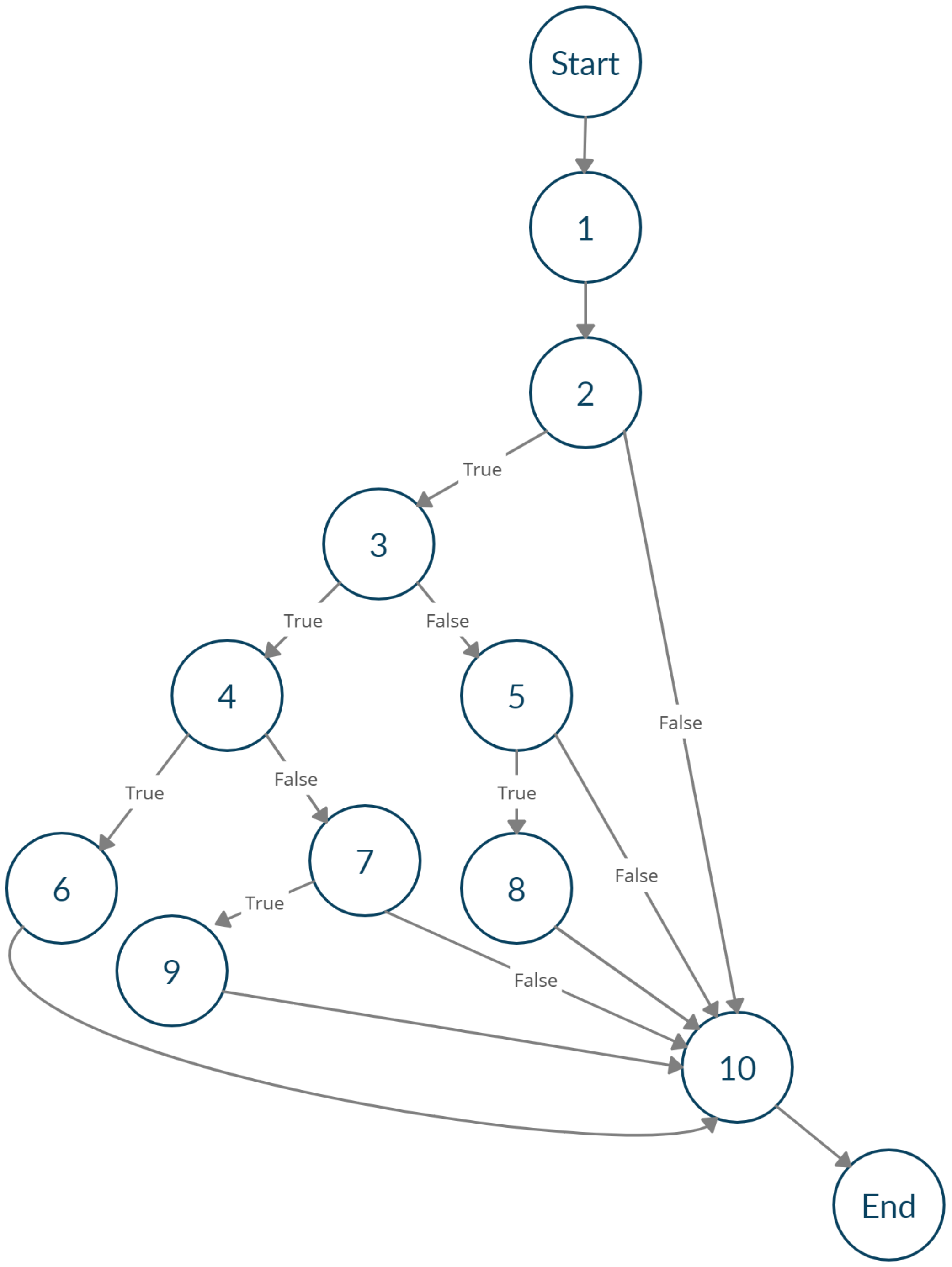
Lean Body Mass Calculator



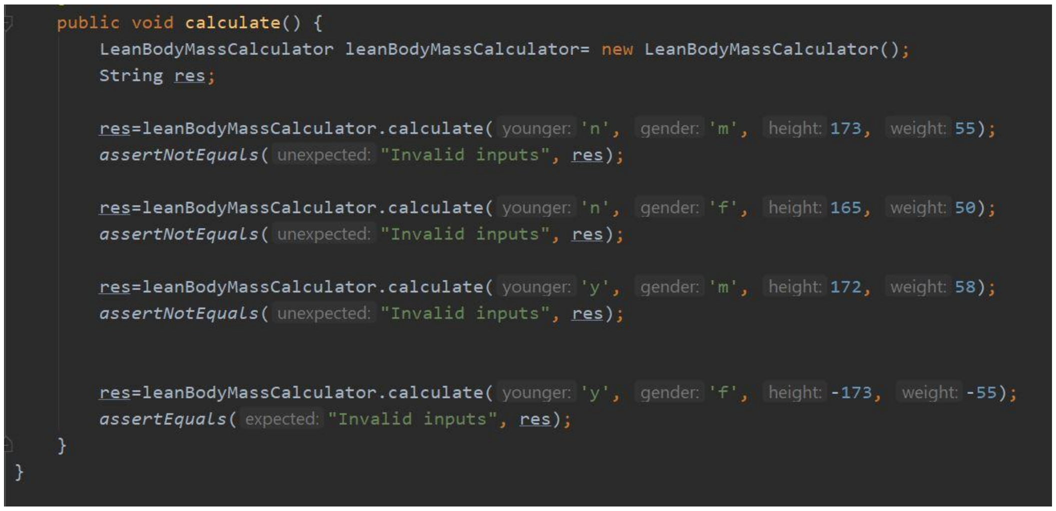
```
1   package com.example.calculator;
2
3   import java.util.Scanner;
4
5   public class LeanBodyMassCalculator {
6       public double calcBodyMass(char younger, char gender, double height, double weight) {
7           double mass = -1;
8           if(height > 0 && weight > 0) {
9               if(younger == 'N' || younger == 'n') {
10                  if(gender == 'M' || gender == 'm')
11                      mass = 0.407*weight + 0.267*height - 19.2;
12                  else if(gender == 'F' || gender == 'f')
13                      mass = 0.252*weight + 0.473*height - 48.3;
14              }
15              else if (younger == 'Y' || younger == 'y') {
16                  double inter_w = Math.pow(weight, 0.6469);
17                  double inter_h = Math.pow(height, 0.7236);
18                  mass = 0.0215*inter_w*inter_h;
19              }
20          }
21          return mass;
22      }
```

Block Number :-

Node Number	Line Numbers (Refer Code)
1	7
2	8
3	9
4	10
5	15
6	11
7	12
8	16,17,18
9	13
10	21


Start
1
2
True
False
3
True
False
4
5
True
False
True
False
6
7
8
True
False
9
False
10
End

Test Cases:-



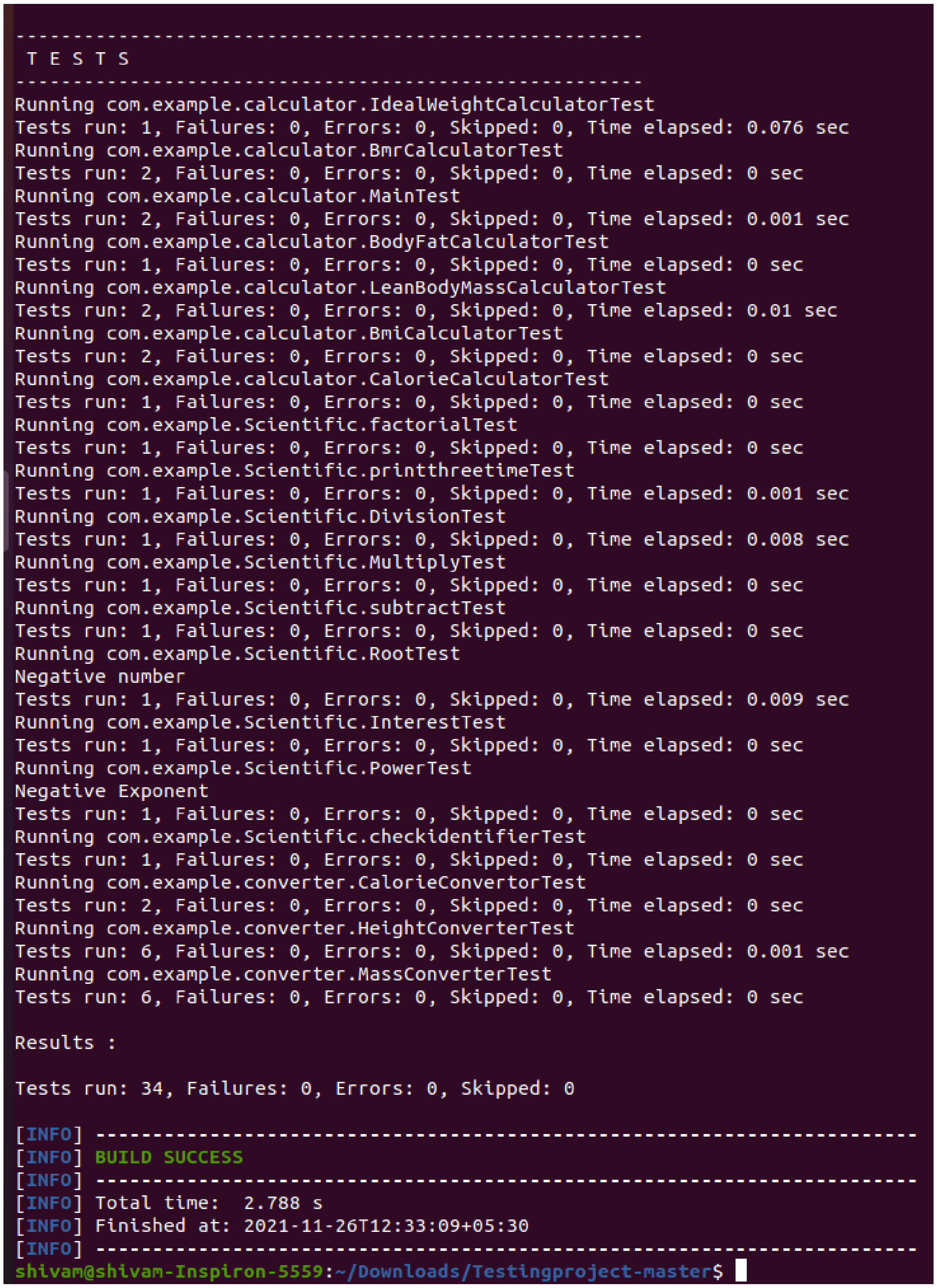
```
    public void calculate() {
        LeanBodyMassCalculator leanBodyMassCalculator= new LeanBodyMassCalculator();
        String res;

        res=leanBodyMassCalculator.calculate( younger: 'n',  gender: 'm',  height: 173,  weight: 55);
        assertNotEquals( unexpected: "Invalid inputs", res);

        res=leanBodyMassCalculator.calculate( younger: 'n',  gender: 'f',  height: 165,  weight: 50);
        assertNotEquals( unexpected: "Invalid inputs", res);

        res=leanBodyMassCalculator.calculate( younger: 'y',  gender: 'm',  height: 172,  weight: 58);
        assertNotEquals( unexpected: "Invalid inputs", res);


        res=leanBodyMassCalculator.calculate( younger: 'y',  gender: 'f',  height: -173,  weight: -55);
        assertEquals( expected: "Invalid inputs", res);
    }
}
```

The above Test cases cove the following path:-

1. [1, 2, 3, 4, 6, 10, end]
2. [1, 2, 3, 4, 7, 9, 10, end]
3. [1, 2, 3, 5, 8, 10, end]
4. [1, 2, 10, end]

These paths cover all the prime paths and edges.

Execution of Test Cases



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.calculator.IdealWeightCalculatorTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.076 sec
Running com.example.calculator.BmrCalculatorTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.calculator.MainTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec
Running com.example.calculator.BodyFatCalculatorTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.calculator.LeanBodyMassCalculatorTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.01 sec
Running com.example.calculator.BmiCalculatorTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.calculator.CalorieCalculatorTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.Scientific.factorialTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.Scientific.printthreetimeTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec
Running com.example.Scientific.DivisionTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 sec
Running com.example.Scientific.MultiplyTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.Scientific.subtractTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.Scientific.RootTest
Negative number
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 sec
Running com.example.Scientific.InterestTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.Scientific.PowerTest
Negative Exponent
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.Scientific.checkidentifierTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.converter.CalorieConvertorTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running com.example.converter.HeightConverterTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec
Running com.example.converter.MassConverterTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec

Results :

Tests run: 34, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.788 s
[INFO] Finished at: 2021-11-26T12:33:09+05:30
[INFO] ------------------------------------------------------------------------
shivam@shivam-Inspiron-5559:~/Downloads/Testingproject-master$ 
```

Team Members Contribution

Shivam Jain - Implemented the source code on which testing is to be performed.

Vaibhav Tandon - Design Test cases for the functions in code based on CFG.

Rahul Vishnoi - Creating Control flow graphs for the functions.