

Software Testing Project

<https://github.com/Gaurav-Tilokani/SavvySolver>

Gaurav Tilokani MT2022045

Jay Parekh MT2022052

Source Code Description

- We have developed a project-specific aptitude solutions functions that combines all major aptitude features.
- It is a console-based java application.

The code provides the following functionalities.

1. It calculates the surface areas and volumes of various 2D and 3D figures like squares, rectangles, circles, cubes, spheres, etc. Therefore, it covers the measurement topic of quantitative aptitude.
2. It calculates and returns the status of profit and loss based on the cost price and selling price provided as inputs.
3. Compound interest and simple interest can also be calculated using it.
4. The program provides a number of core features such as finding the nth term, printing the first n terms, computing the sum of the first n terms, etc, for progressions such as Arithmetic Progressions (AP) and Geometric Progressions (GP).
5. Additionally, we can compute percentage changes, percentage values for fractions, and the number of days it will take if people with given efficiencies cooperate.

Testing strategy

Data flow graph:-

- For each function, we are first creating a Control flow graph from code.
- We are creating dfg.
- After creating dfg we are finding du pairs, du paths, and All du path coverage with the help of <https://cs.gmu.edu:8443/offutt/coverage/DFGraphCoverage>.
- Based on all du paths we are creating test cases and then testing using Junit.

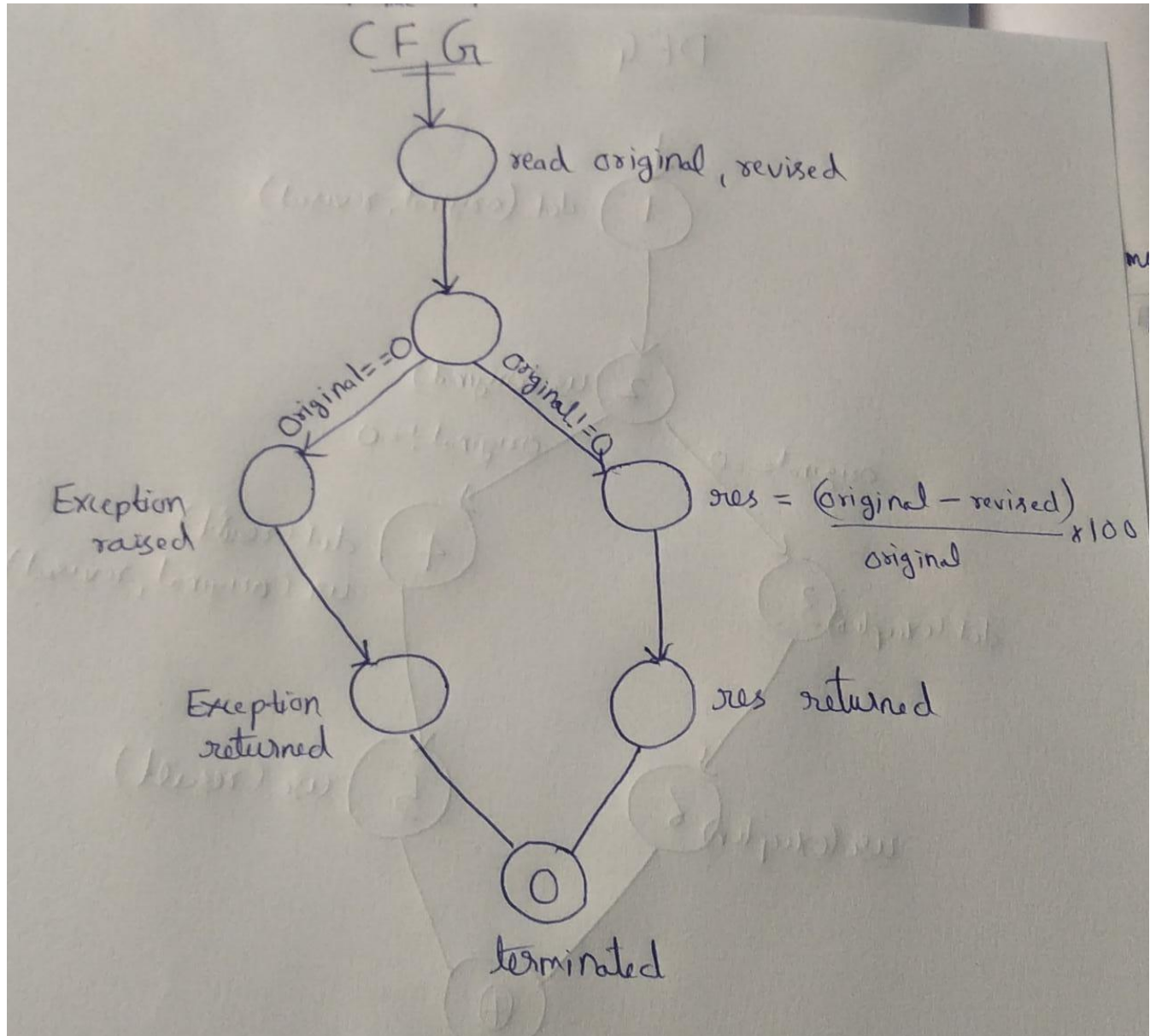
Tools used:-

- Junit for test cases
- <https://cs.gmu.edu:8443/offutt/coverage/DFGraphCoverage> for TR generation.
- Below are some of the functions among the ones we have tested.

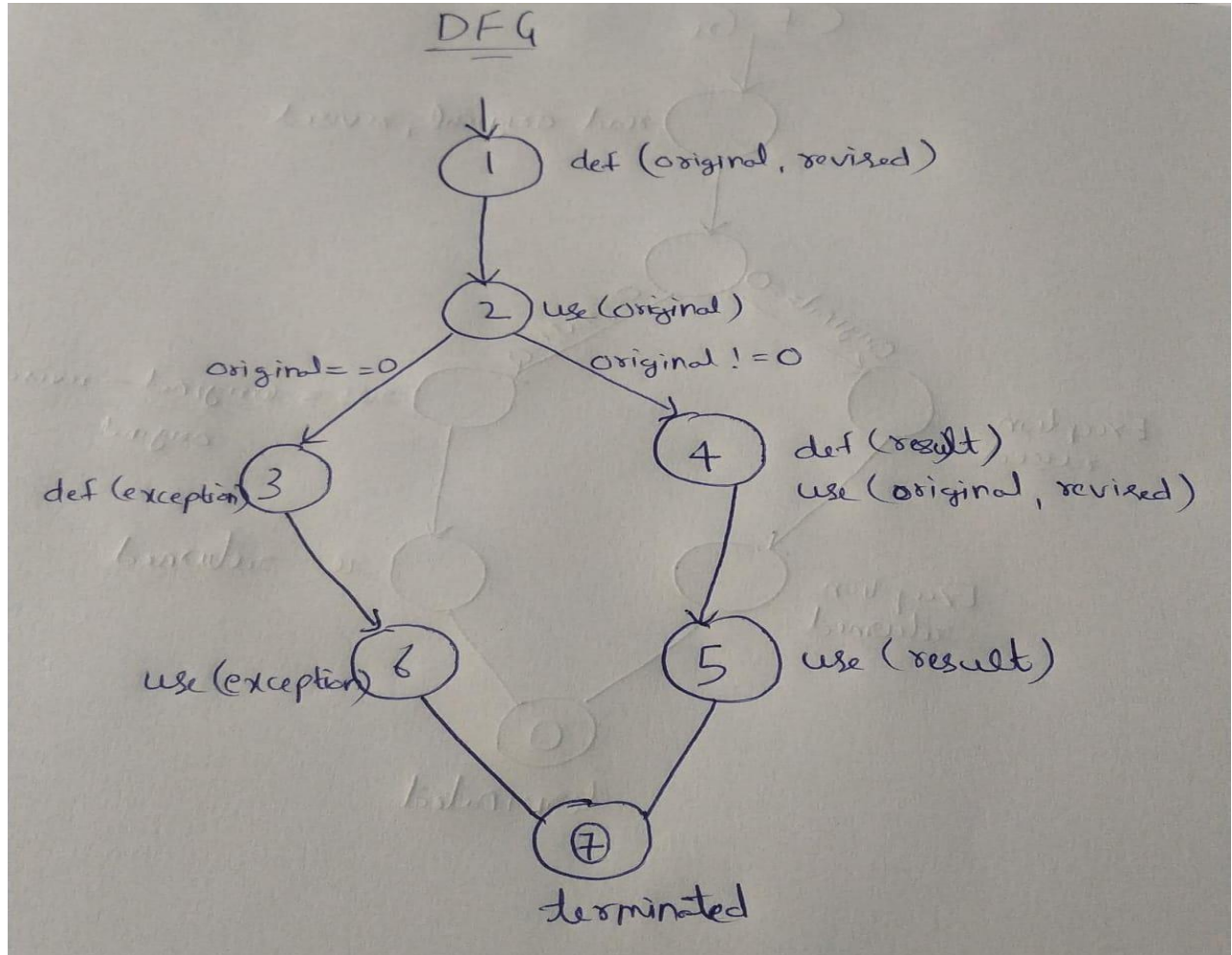
1. Percentage Change:-

```
public String percentageChange(double original, double revised)
{
    try {
        if (original == 0) {
            throw new Exception();
        }
        double res = ((revised - original) / original) * 100;
        return Double.toString(res);
    } catch (Exception e)
    {
        return "Invalid original value";
    }
}
```

CFG



DFG

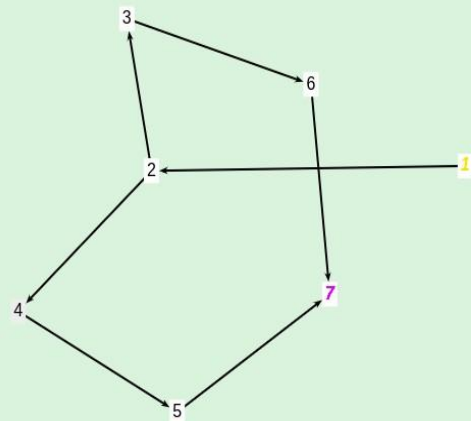


Graph, DU Pairs, Du Paths, and All du path coverage

DU Pairs for all variables are:

| Variable | DU Pairs |
|-----------|----------------|
| original | [1,2] [1,4] |
| revised | [1,4] |
| exception | [3,6] |
| result | [4,5] |

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|-----------|------------------|
| original | [1,2] [1,2,4] |
| revised | [1,2,4] |
| exception | [3,6] |
| result | [4,5] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|-----------|----------------------------|
| original | [1,2,3,6,7] [1,2,4,5,7] |
| revised | [1,2,4,5,7] |
| exception | [1,2,3,6,7] |
| result | [1,2,4,5,7] |

We have total 2 unique paths here:-

1. [1,2,3,6,7]
Input:- original = 0, revised = 10
Output:- Invalid original value.
2. [[1,2,4,5,7]
Input:- original = 10, revised = 15
Output:- 50.0

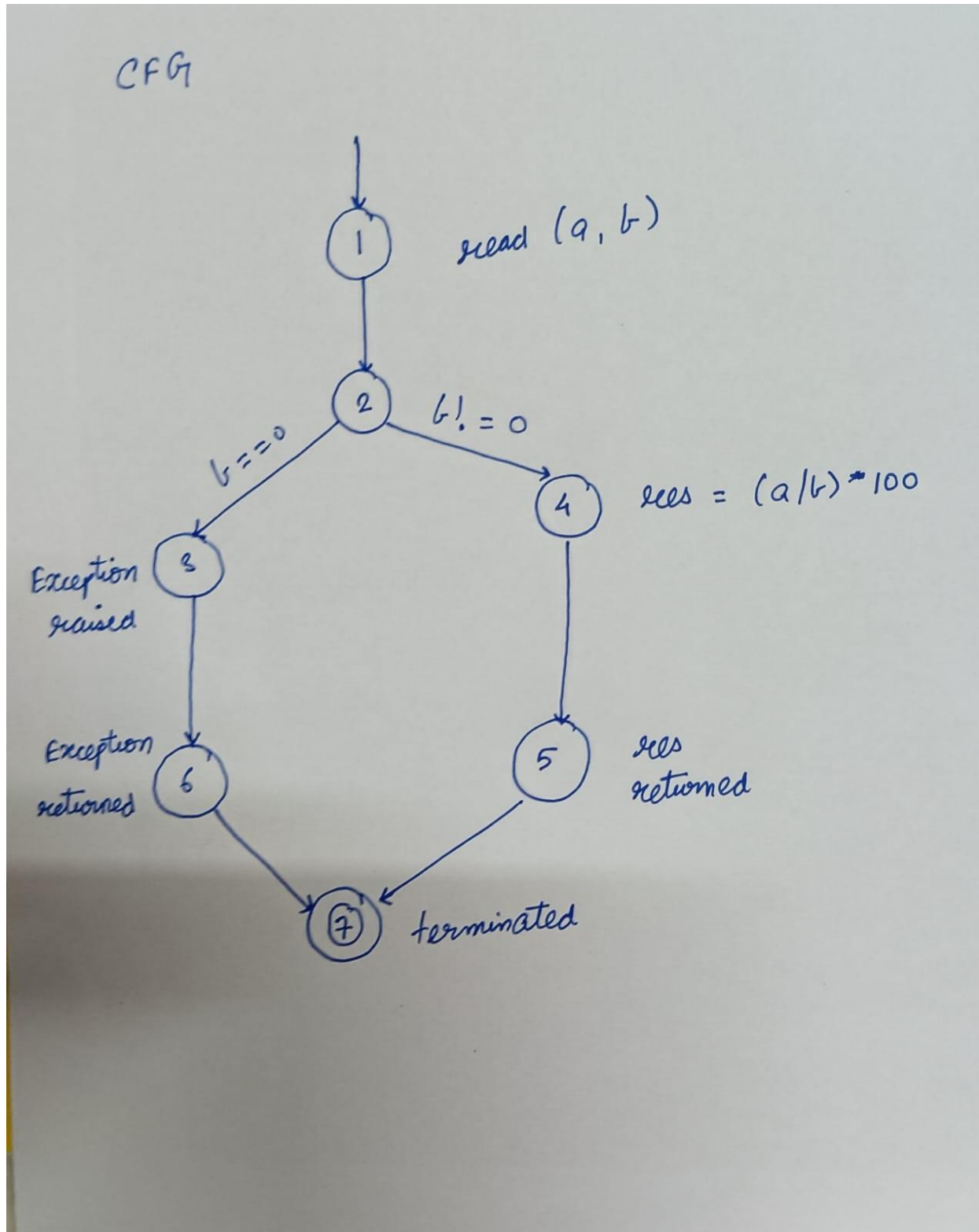
Result

```
✓ percentageChange()=>follows path [1,2,3,6,7] 2 ms
✓ percentageChange()=>follows path [1,2,4,5,7] 2 ms
```

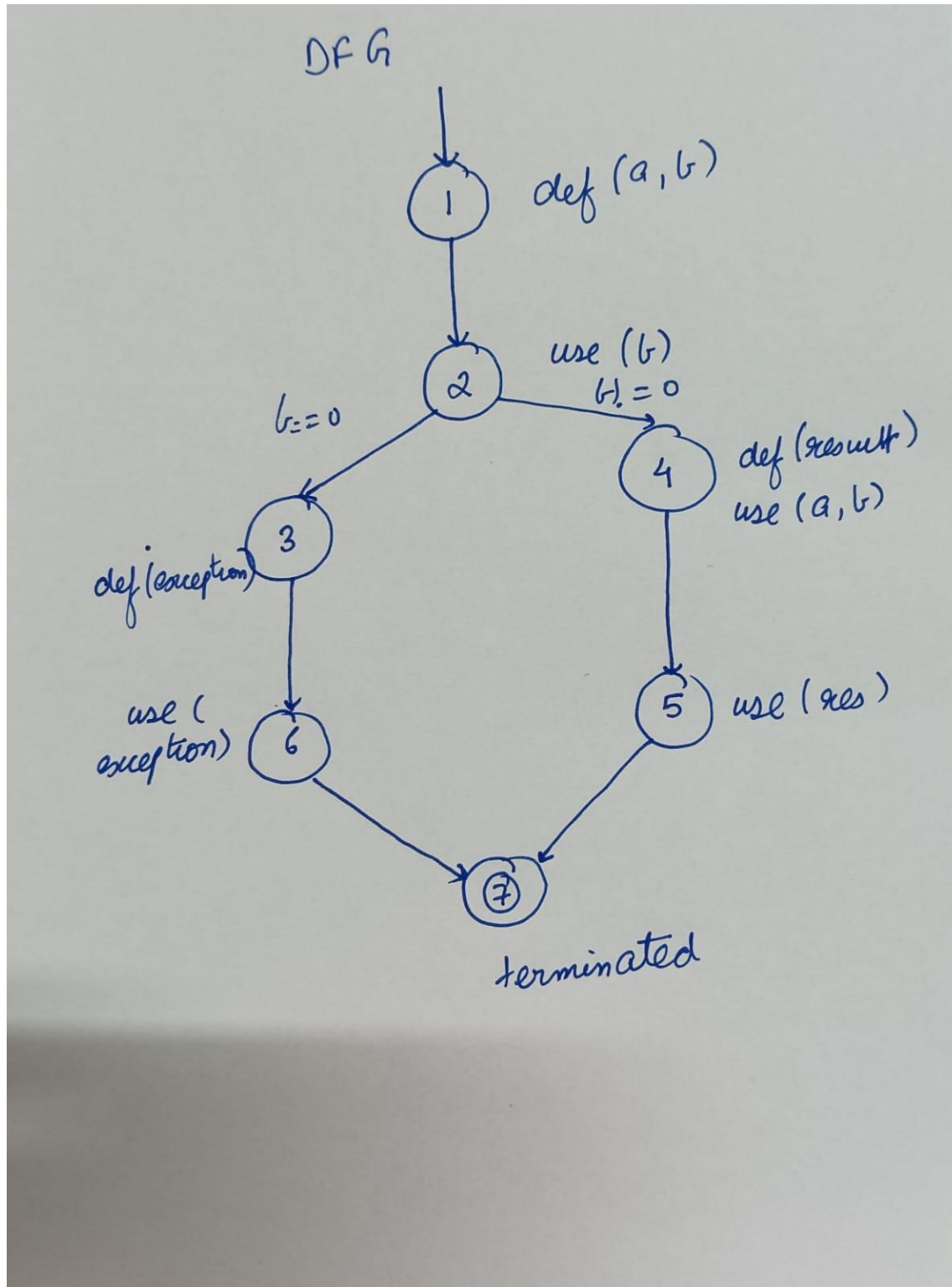
2. Fraction to percentage converter:-

```
public String fractionToPercentageConverter(double a, double b)
{
    try
    {
        if(b==0)
        {
            throw new Exception();
        }
        else
        {
            double res = (a/b)*100;
            return Double.toString(res);
        }
    }catch(Exception e)
    {
        return "Value of b cannot be zero";
    }
}
```

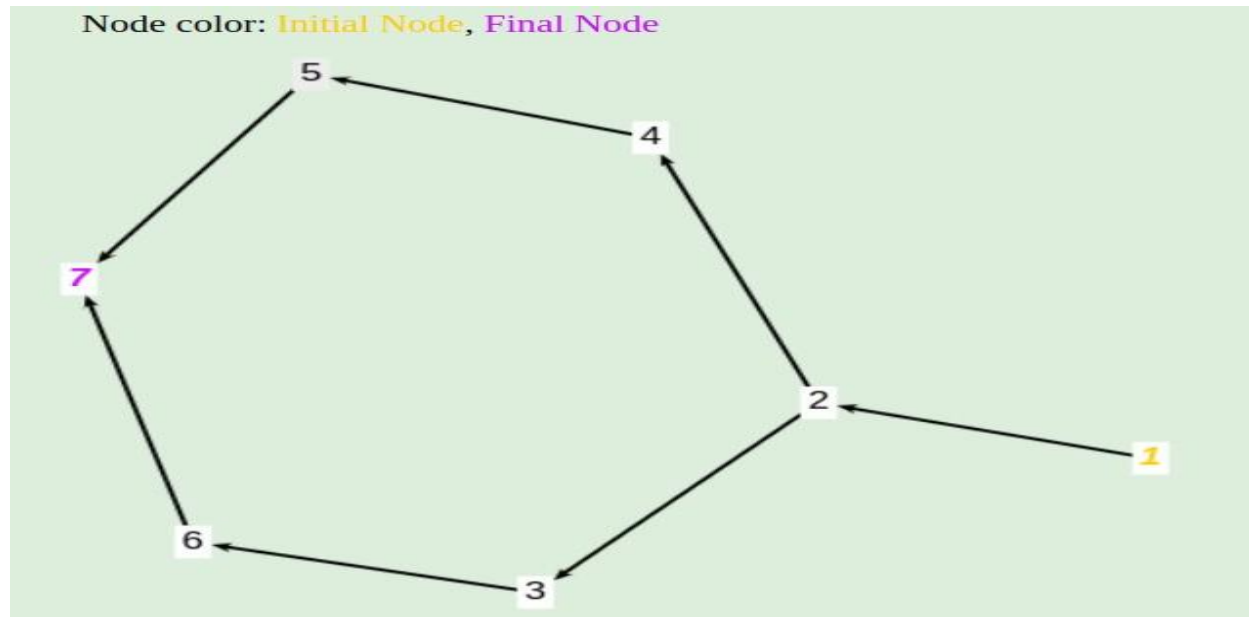

CFG



DFG



Graph, DU Pairs, Du Paths and All du path coverage



DU Paths for all variables are:

| Variable | DU Paths |
|-----------|------------------|
| a | [1,2,4] |
| b | [1,2] [1,2,4] |
| exception | [3,6] |
| res | [4,5] |

DU Pairs for all variables are:

| Variable | DU Pairs |
|-----------|----------------|
| a | [1,4] |
| b | [1,2] [1,4] |
| exception | [3,6] |
| res | [4,5] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|-----------|----------------------|
| a | [1,2,4,5,7] |
| b | [1,2,4,5,7] |
| exception | [1,2,3,6,7] |
| res | [1,2,4,5,7] |

We have total 2 unique paths here:-

1. [1,2,3,6,7]
Input:- a = 10, b = 0
Output:- value of b cannot be zero
2. [[1,2,4,5,7]
Input:- original = 20, revised = 40
Output:- 50.0

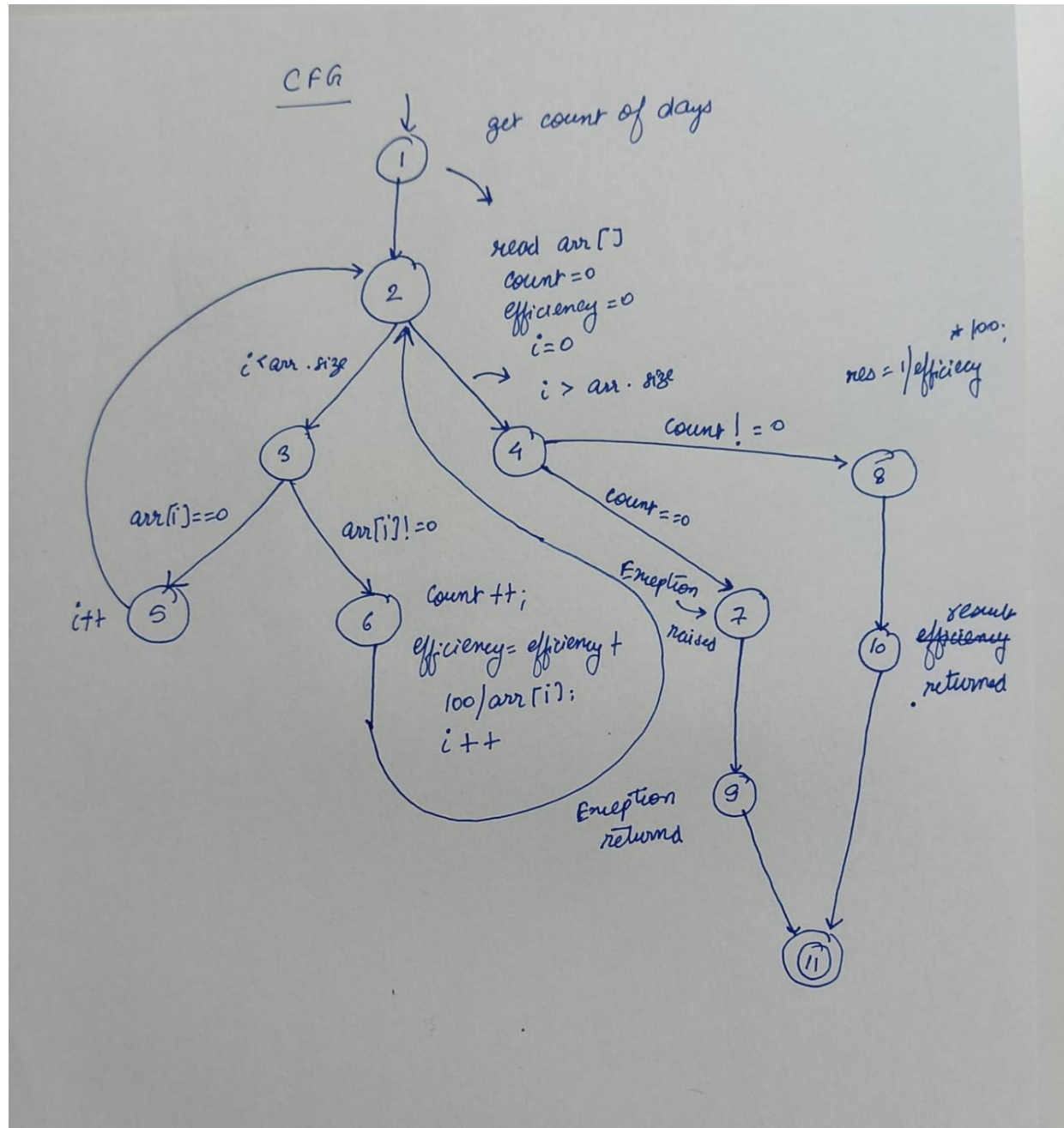
Result

```
✓ fractionToPercentageConverter()=>follows path [1,2,3,6,7] 4 ms
✓ fractionToPercentageConverter()=>follows path [1,2,4,6,7] 11 ms
```

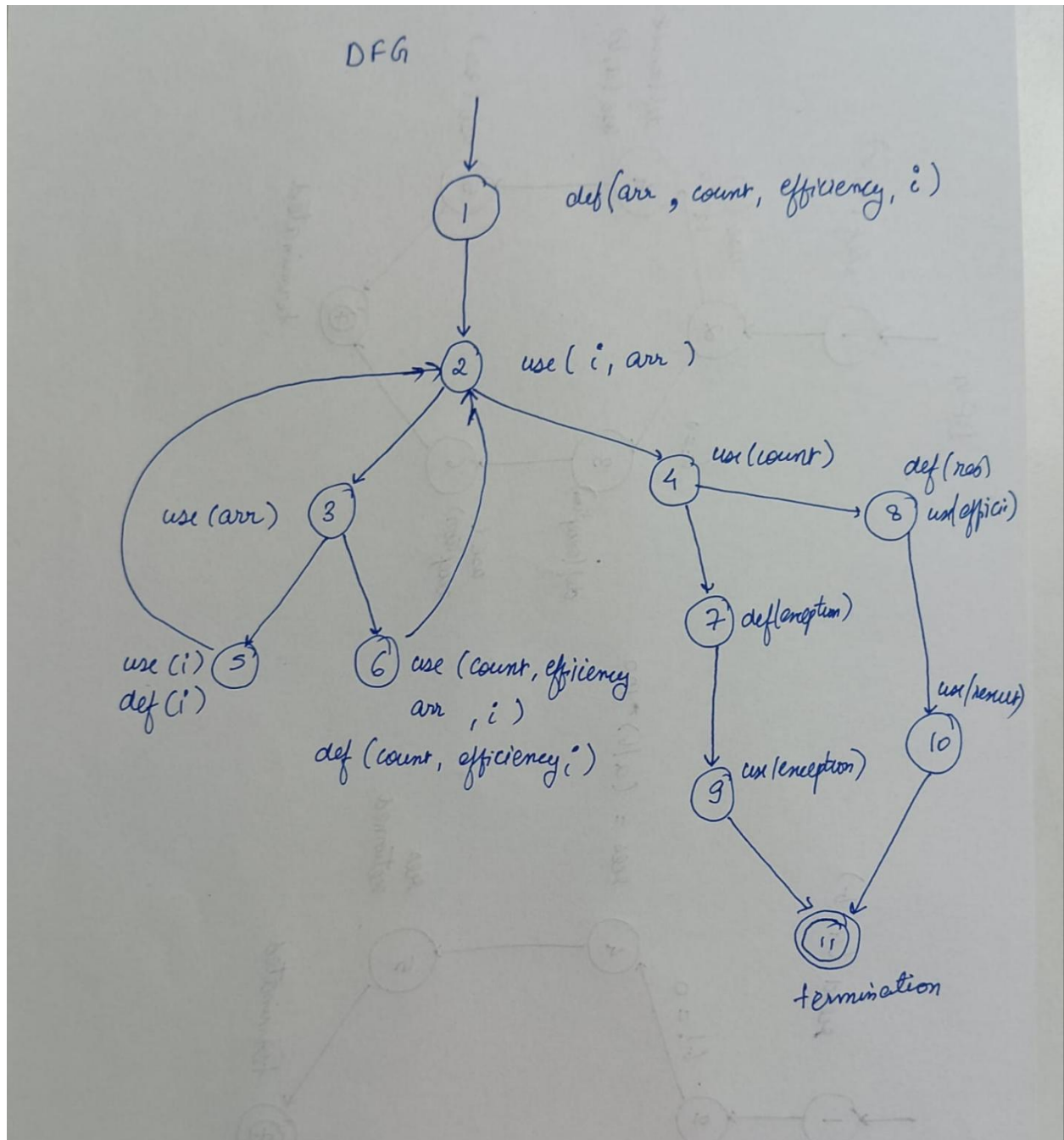
3. Count of Days:-

```
public String getCountOfDays(int[] arr)
{
    int count = 0;
    double efficiency=0;
    for(int i=0;i<arr.length;i++)
    {
        if(arr[i]!=0)
        {
            count++;
            efficiency = efficiency + 100/arr[i];
        }
    }
    try
    {
        if(count==0)
        {
            throw new Exception();
        }
        double res = (1/efficiency)*100;
        return Double.toString(res);
    }catch (Exception e)
    {
        return "Invalid input";
    }
}
```

CFG

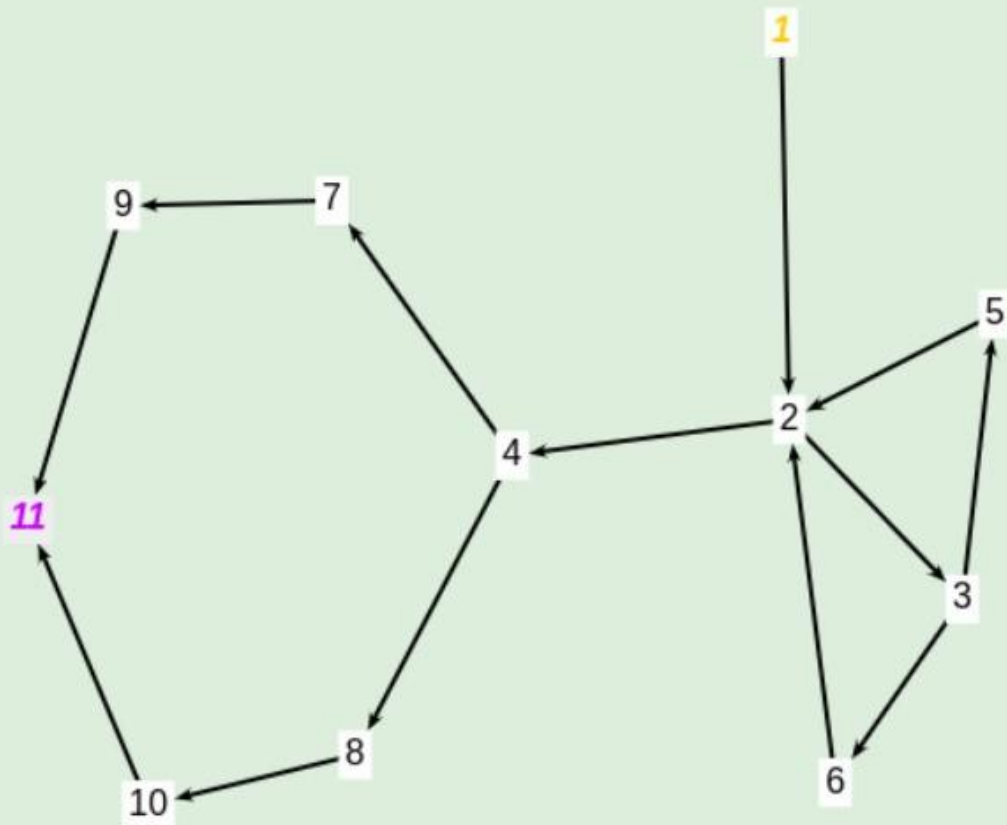


DFG



Graph, DU Pairs, Du Paths and All du path coverage

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|------------|---|
| arr | [1,2] [1,2,3] [1,2,3,6] |
| count | [1,2,4] [1,2,3,6] [6,2,4] [6,2,3,6] |
| efficiency | [1,2,4,8] [1,2,3,6] [6,2,4,8] [6,2,3,6] |
| i | [1,2] [1,2,3,5] [1,2,3,6] [5,2] [5,2,3,6] [5,2,3,5] [6,2] [6,2,3,6] [6,2,3,5] |
| exception | [7,9] |
| result | [8,10] |

DU Pairs for all variables are:

| Variable | DU Pairs |
|------------|---|
| arr | [1,2] [1,3] [1,6] |
| count | [1,6] [1,4] [6,6] [6,4] |
| efficiency | [1,6] [1,8] [6,6] [6,8] |
| i | [1,2] [1,5] [1,6] [5,2] [5,5] [5,6] [6,2] [6,5] [6,6] |
| exception | [7,9] |
| result | [8,10] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|------------|--|
| arr | [1,2,4,7,9,11] [1,2,3,5,2,4,7,9,11] [1,2,3,6,2,4,7,9,11] |
| count | [1,2,4,7,9,11] [1,2,3,6,2,4,7,9,11] [1,2,3,6,2,3,6,2,4,7,9,11] |
| efficiency | [1,2,4,8,10,11] [1,2,3,6,2,4,7,9,11] [1,2,3,6,2,4,8,10,11] [1,2,3,6,2,3,6,2,4,7,9,11] |
| i | [1,2,4,7,9,11] [1,2,3,5,2,4,7,9,11] [1,2,3,6,2,4,7,9,11] [1,2,3,5,2,3,6,2,4,7,9,11] [1,2,3,5,2,3,5,2,4,7,9,11] [1,2,3,6,2,3,6,2,4,7,9,11] [1,2,3,6,2,3,5,2,4,7,9,11] |
| exception | [1,2,4,7,9,11] |
| result | [1,2,4,8,10,11] |

Here we have total 3 unique paths:-

1. [1,2,3,5,2,4,7,9,11]

Input:- arr[]={}

Output:-Invalid input

2. [1,2,3,6,2,4,7,9,11]

Input:- arr[]={1}

Output:- 1.0

3. [1,2,4,7,9,11]

Input:- arr[]={}

Output:- Invalid input

Result

```

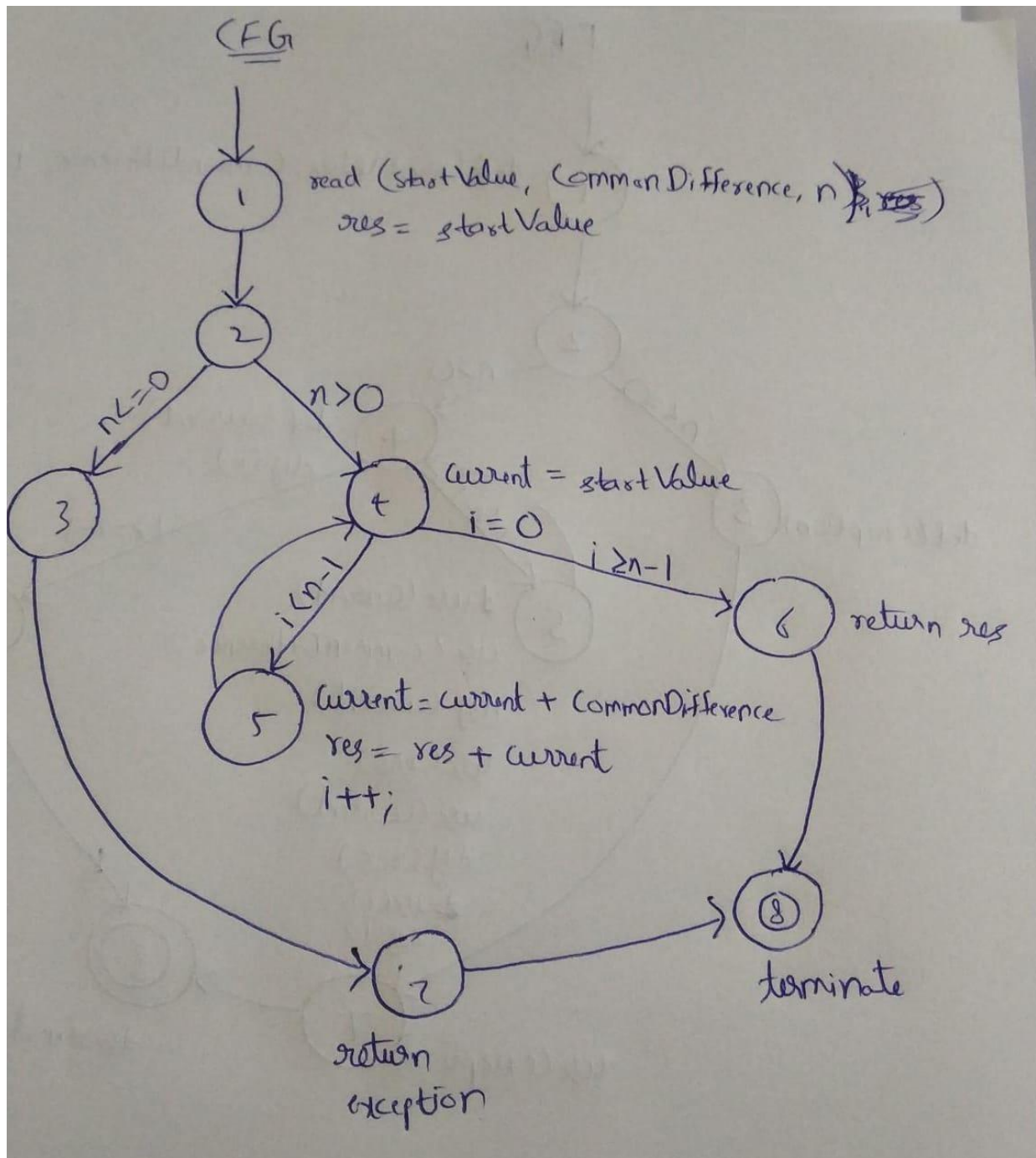
✓ getCountOfDays()=>follows path [1,2,3,5,2,4,7,9,11]      1 ms
✓ getCountOfDays()=>follows path [1,2,3,6,2,4,7,9,11]      2 ms
✓ getCountOfDays()=>follows path [1,2,4,7,9,11]            2 ms

```

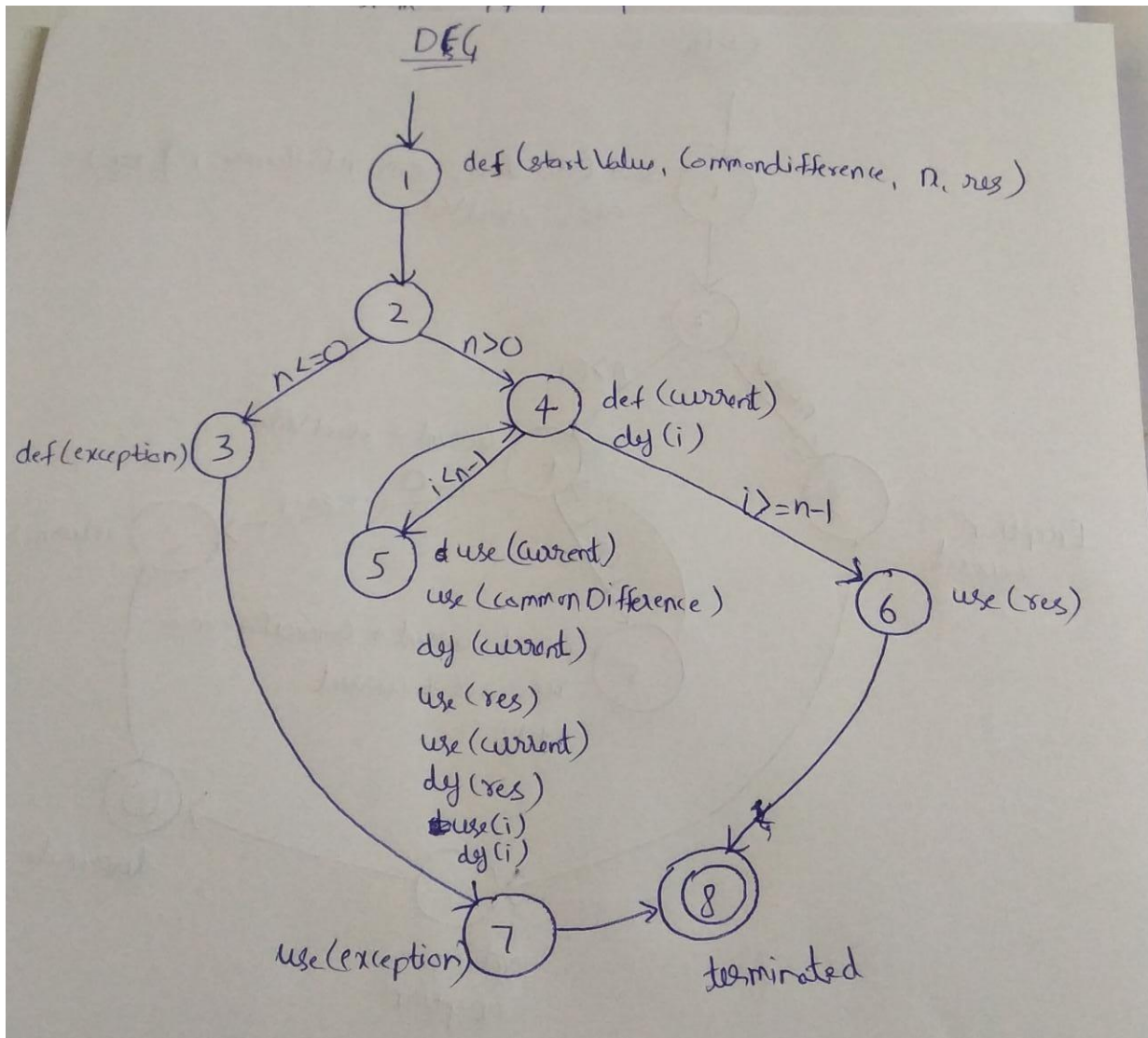
4. Find the first N terms of AP

```
public String findFirstNTermsAP(double startValue, double commonDifference, int n)
{
    String res = Double.toString(startValue);
    try
    {
        if(n<=0)
        {
            throw new Exception();
        }
        double current = startValue;
        for(int i=0;i<n-1;i++)
        {
            current = current + commonDifference;
            res = res + "," + Double.toString(current);
        }
        return res;
    }catch(Exception e)
    {
        return "Invalid count input";
    }
}
```

CFG



DFG

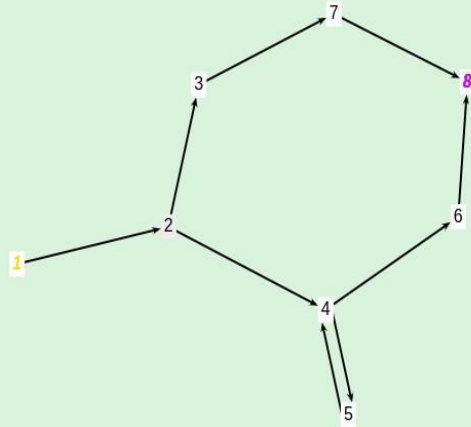


Graph, DU Pairs, Du Paths and All du path coverage

DU Pairs for all variables are:

| Variable | DU Pairs |
|------------------|----------------------------------|
| startValue | [1,1] [1,4] |
| commonDifference | [1,5] |
| n | [1,2] [1,4] |
| res | [1,5] [1,6] [5,5] [5,6] |
| exception | [3,7] |
| current | [4,5] [5,5] |
| i | [4,4] [4,5] [5,4] [5,5] |

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|------------------|--|
| startValue | [1,2,4] |
| commonDifference | [1,2,4,5] |
| n | [1,2] [1,2,4] |
| res | [1,2,4,6] [1,2,4,5] [5,4,6] [5,4,5] |
| exception | [3,7] |
| current | [4,5] |
| i | [4,5] [5,4] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|------------------|---|
| startValue | [1,2,4,6,8] |
| commonDifference | [1,2,4,5,4,6,8] |
| n | [1,2,3,7,8] [1,2,4,6,8] |
| res | [1,2,4,6,8] [1,2,4,5,4,6,8] [1,2,4,5,4,5,4,6,8] |
| exception | [1,2,3,7,8] |
| current | [1,2,4,5,4,6,8] |
| i | [1,2,4,5,4,6,8] |

Here we have 3 unique paths:-

1. [1,2,3,7,8]

Input:- startValue=1, commonDifference=3, n=-1

Output:- Invalid count Input

2. [1,2,4,6,8]

Input:- startValue=1, commonDifference=3, n=1

Output:- 1.0

3. [1,2,4,5,4,5,4,6,8]

Inputs:- startValue=1, commonDifference=3, n=3

Output:- 1.0, 4.0, 7.0

Result:-

```
✓ findFirstNTermsAP()=>follows path [1,2,3,7,8] 72 ms
✓ findFirstNTermsAP()=>follows path [1,2,4,6,8] 2 ms
✓ findFirstNTermsAP()=>follows path [1,2,4,5,4,5,4,6,8] 41 ms
```

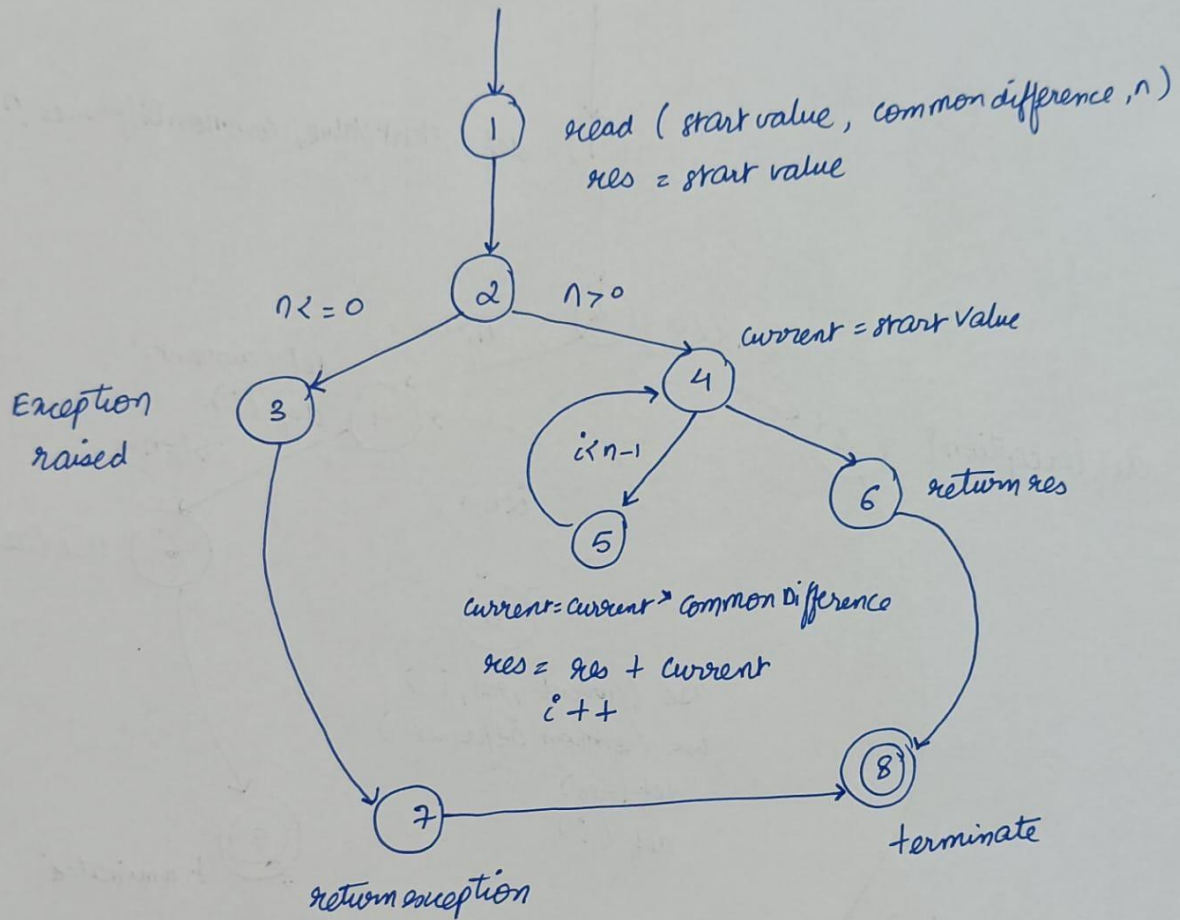

5. Find the First nth Terms of GP

```
public String findFirstNTermsGP(double startValue, double commonDifference, int n)
{
    String res = Double.toString(startValue);
    try
    {
        if(n<=0)
        {
            throw new Exception();
        }
        double current = startValue;
        for(int i=0;i<n-1;i++)
        {
            current = current*commonDifference;
            res = res + "," + Double.toString(current);
        }
        return res;
    }catch(Exception e)
    {
        return "Invalid value of n";
    }
}
```

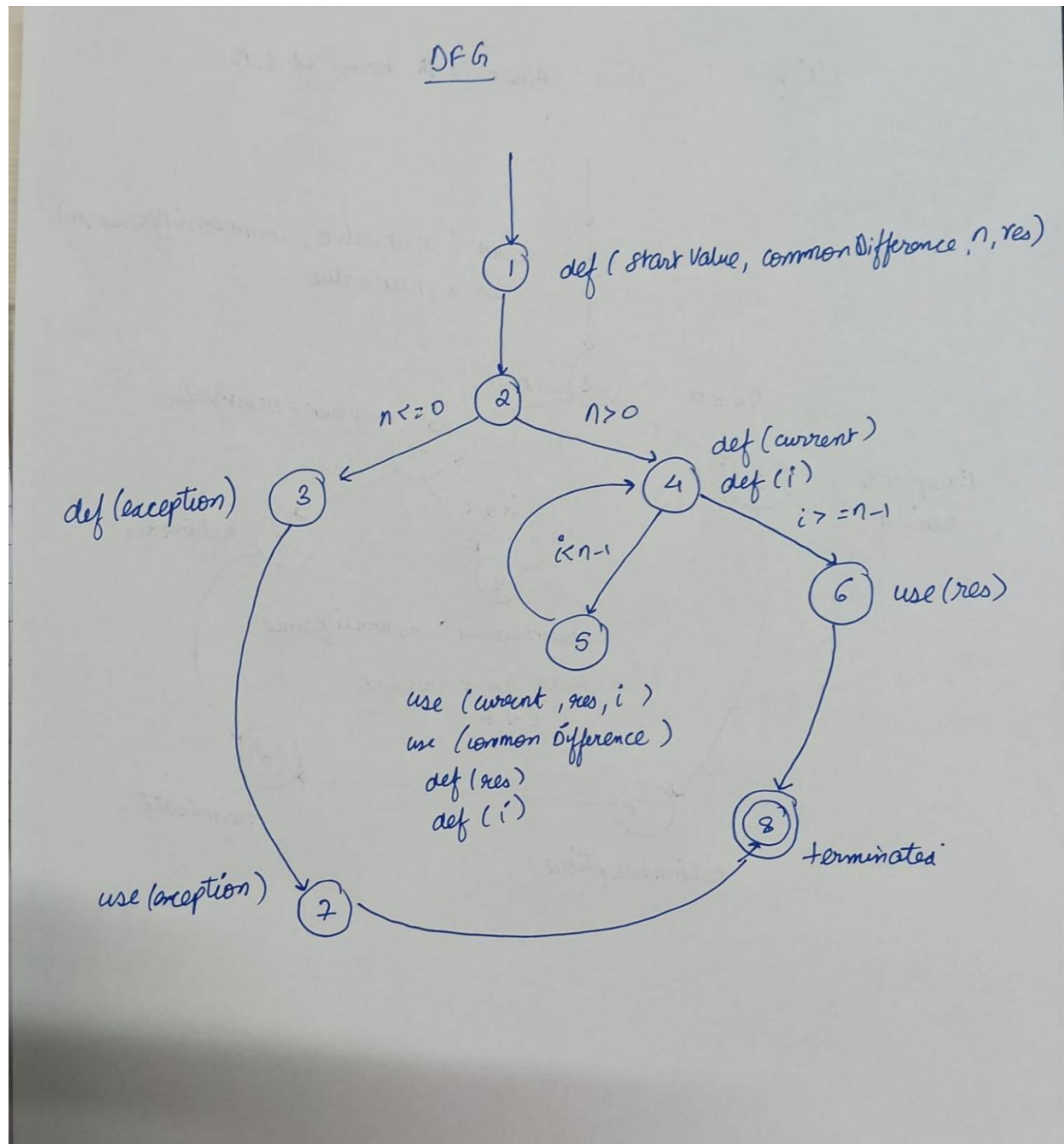

CFG

CFG

Find first N th terms of G.P.

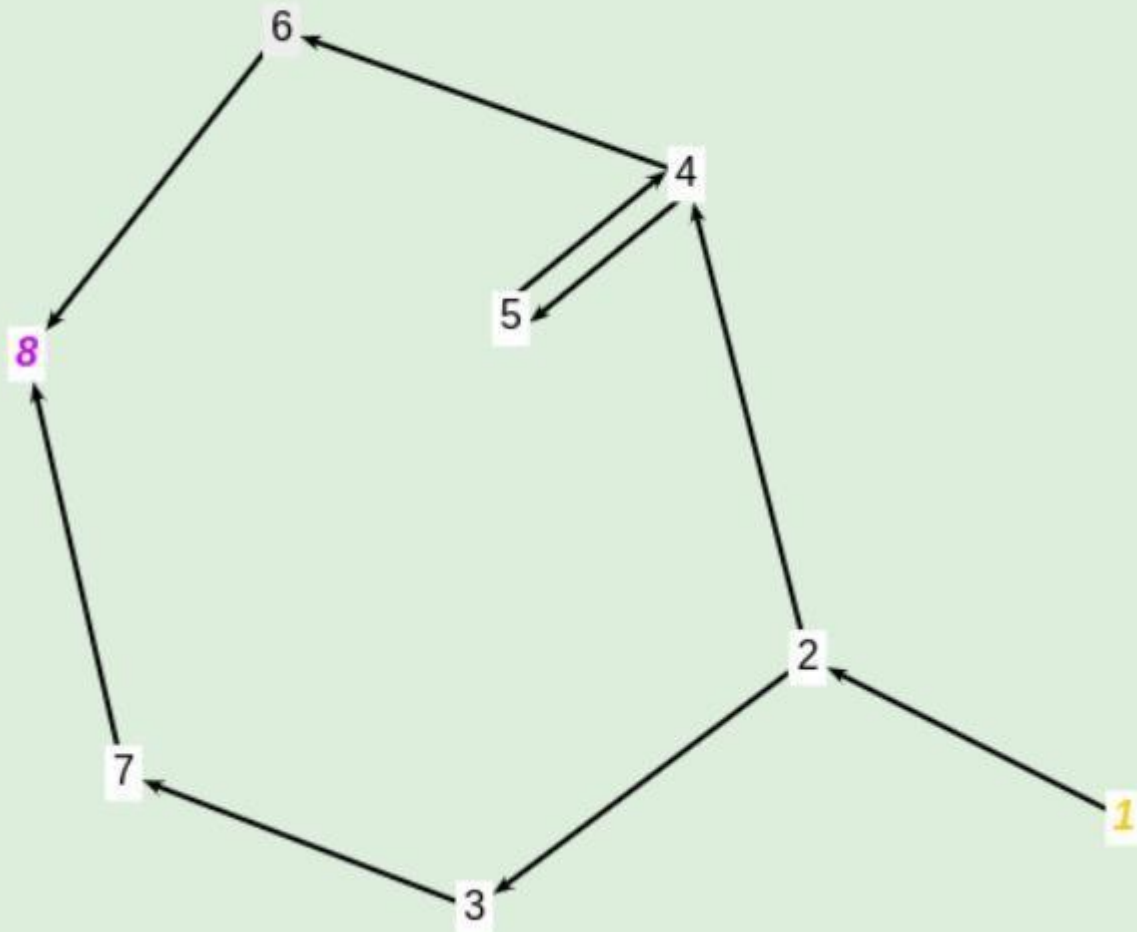


DFG



Graph, DU Pairs, Du Paths and All du path coverage

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|------------------|-----------|
| startvalue | [1,2,4] |
| commonDifference | [1,2,4,5] |
| n | [1,2] |
| current | [4,5] |
| i | [4,5] |
| res | [1,2,4,6] |
| | [1,2,4,5] |
| | [5,4,6] |
| | [5,4,5] |
| exception | [3,7] |

DU Pairs for all variables are:

| Variable | DU Pairs |
|------------------|----------------------------------|
| startvalue | [1,4] |
| commonDifference | [1,5] |
| n | [1,2] |
| current | [4,5] [5,5] |
| i | [4,5] [5,5] |
| res | [1,6] [1,5] [5,6] [5,5] |
| exception | [3,7] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|------------------|---|
| startvalue | [1,2,4,6,8] |
| commonDifference | [1,2,4,5,4,6,8] |
| n | [1,2,4,6,8] |
| current | [1,2,4,5,4,6,8] |
| i | [1,2,4,5,4,6,8] |
| res | [1,2,4,6,8] [1,2,4,5,4,6,8] [1,2,4,5,4,5,4,6,8] |
| exception | [1,2,3,7,8] |

Here we have 3 unique paths:-

1. [1,2,4,5,4,6,8]

Input:- startValue=10, commonDifference=10, n=3

Output:- 10.0, 100.0, 1000.0

2. [1,2,4,6,8]

Input:- startValue=10, commonDifference=10, n=1

Output:- 10.0

3. [1,2,3,7,8]

Input:- startValue=10, commonDifference=10, n=0

Output:- Invalid value of n

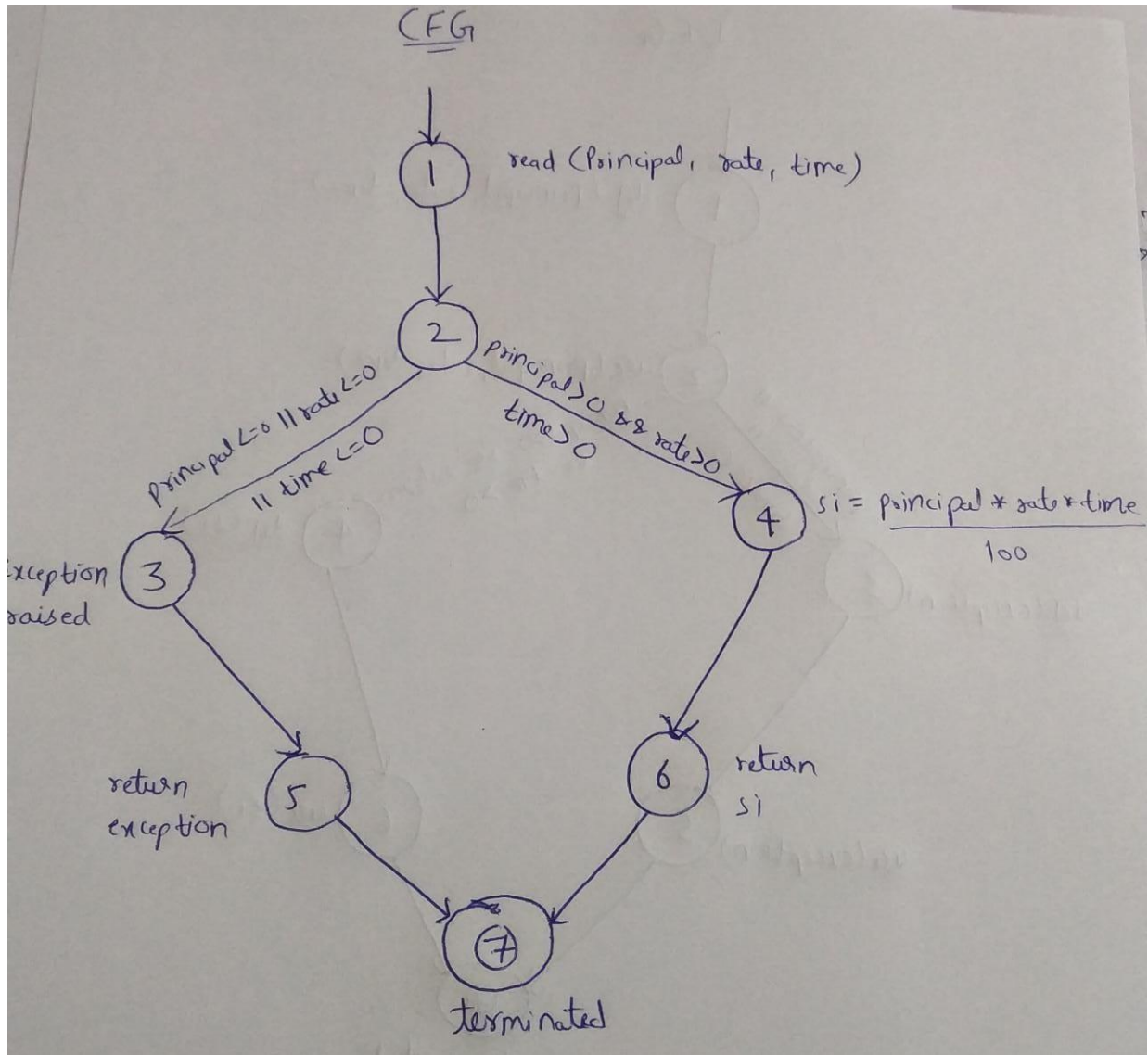
Result

| | |
|---|-------|
| ✓ findFirstNTermsGP()=>follows path [1,2,3,7,8] | 4 ms |
| ✓ findFirstNTermsGP()=>follows path [1,2,4,5,4,6,8] | 10 ms |
| ✓ findFirstNTermsGP()=>follows path [1,2,4,6,8] | 4 ms |

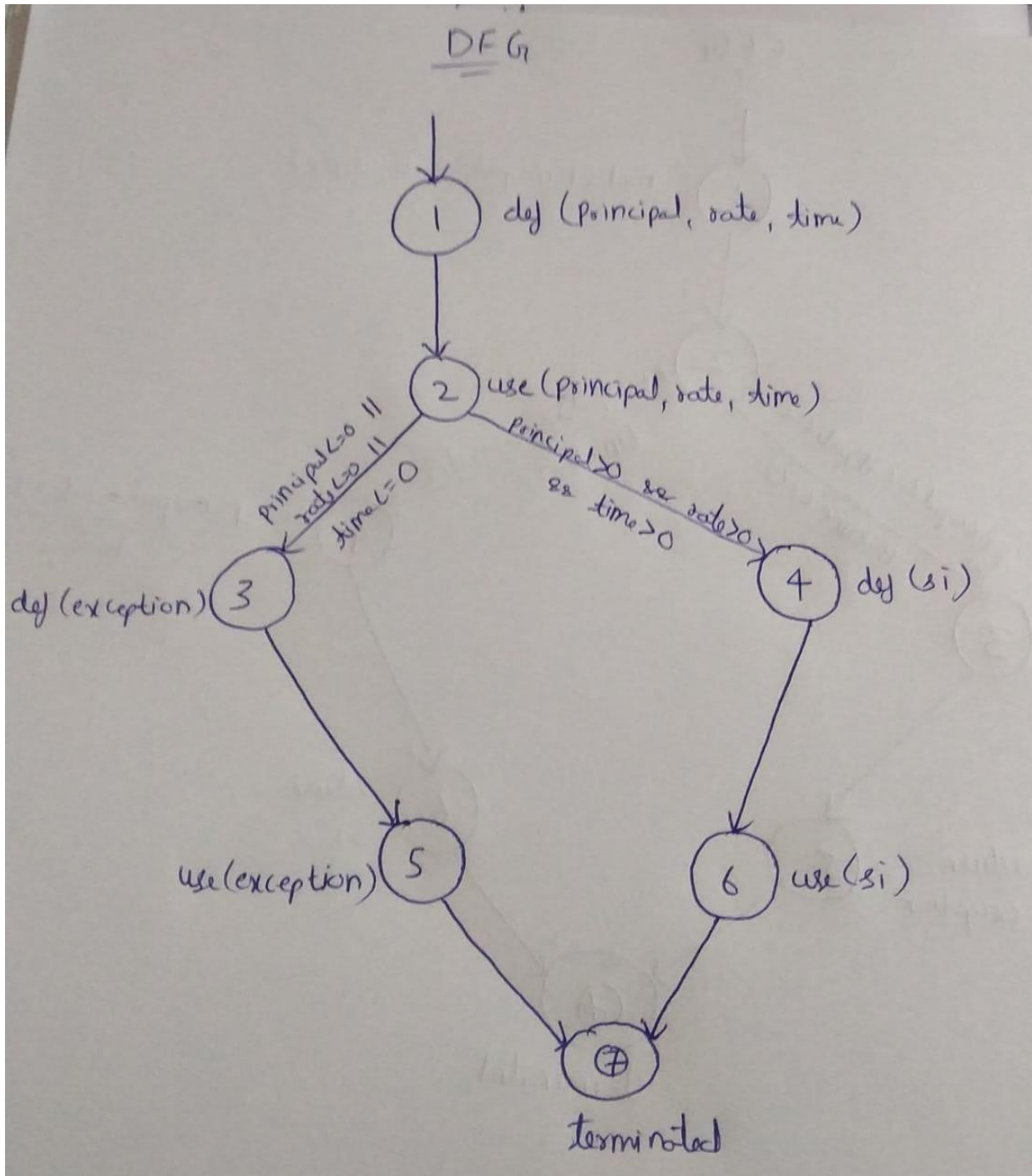
6. Find Simple Interest:-

```
public String simpleInterest(double principal, double rate, double time)
{
    try {
        if (principal <= 0 || rate<=0 || time<=0) {
            throw new Exception();
        }
        double si = (principal * rate * time)/ 100;
        return Double.toString(si);
    }catch (Exception e)
    {
        return "Either principal value or rate value or time value is invalid";
    }
}
```

CFG



DFG

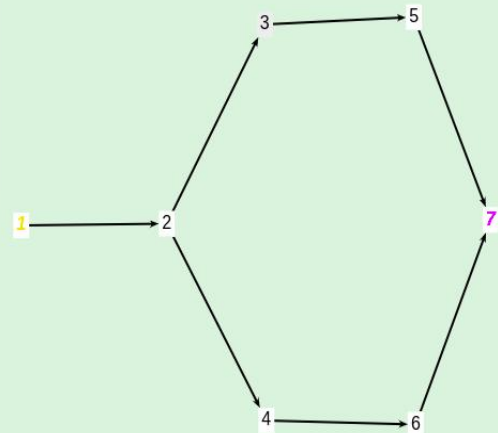


Graph, DU Pairs, Du Paths and All du path coverage

DU Pairs for all variables are:

| Variable | DU Pairs |
|-----------|----------------|
| principal | [1,2] [1,4] |
| rate | [1,2] [1,4] |
| time | [1,2] [1,4] |
| si | [4,6] |
| exception | [3,5] |

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|-----------|------------------|
| principal | [1,2] [1,2,4] |
| rate | [1,2] [1,2,4] |
| time | [1,2] [1,2,4] |
| si | [4,6] |
| exception | [3,5] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|-----------|----------------------------|
| principal | [1,2,3,5,7] [1,2,4,6,7] |
| rate | [1,2,3,5,7] [1,2,4,6,7] |
| time | [1,2,3,5,7] [1,2,4,6,7] |
| si | [1,2,4,6,7] |
| exception | [1,2,3,5,7] |

Here we have two unique paths:-

1. [1,3,5,7]

Input:- principal=10, rate=2, time=0

Output:- Either the value of principal or the value of rate or the value of time is invalid.

2. [1,2,4,6,8]

Input:- principal=100, rate=10, time=2

Output:- 20.0

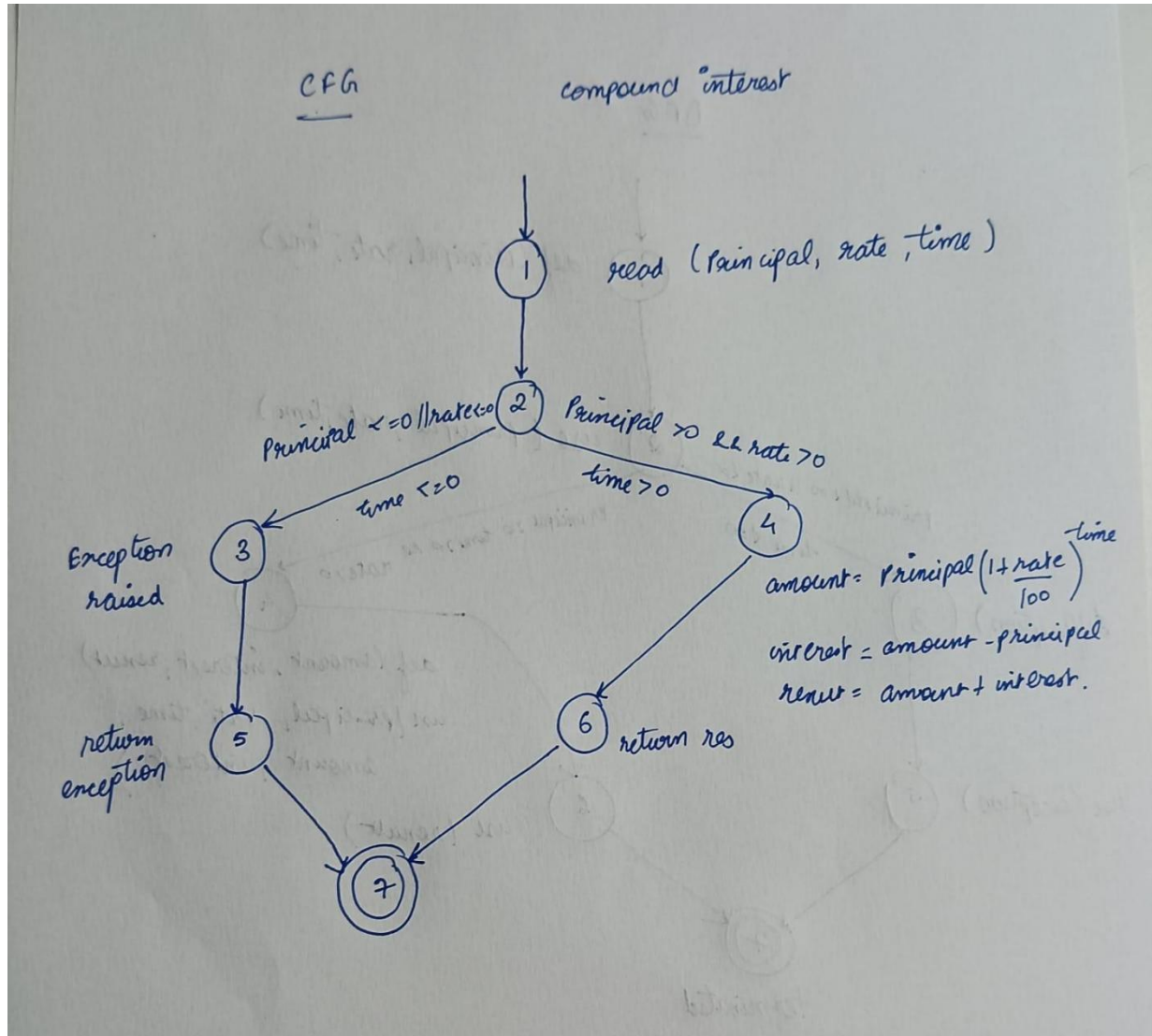
Result:-

```
✓ simpleInterest()=>follows path [1,3,5,7] 2 ms
✓ simpleInterest()=>follows path [1,2,4,6,7] 3 ms
```

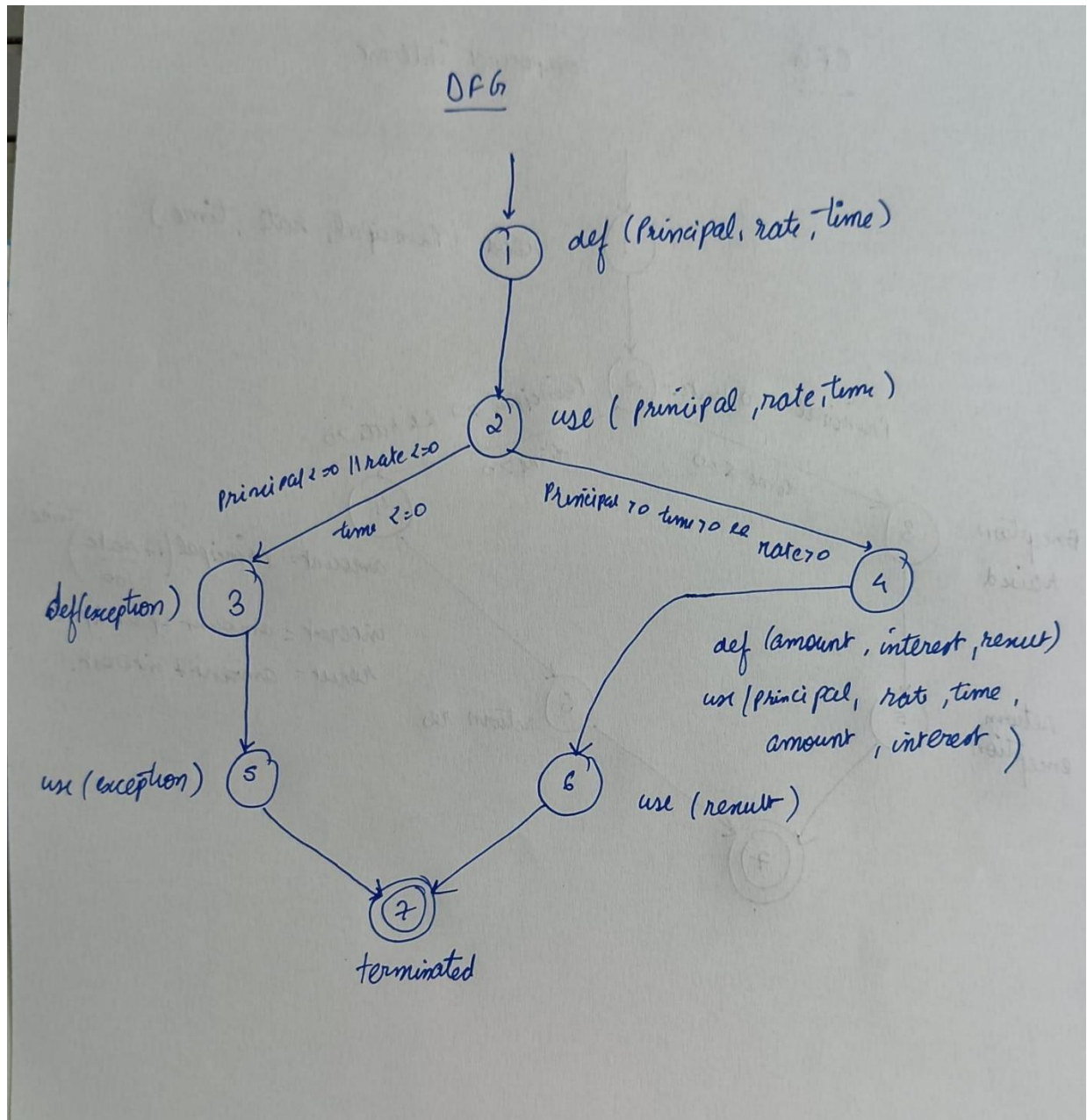
7. Compound Interest :-

```
public String compoundInterest(double principal, double rate, double time)
{
    String result;
    try {
        if (principal == 0 || rate==0 || time==0) {
            throw new Exception();
        }
        double amount = principal * (Math.pow((1 + rate/100), (time)));
        double interest = amount - principal;
        return result = "Amount is: " + amount + " Interest is: " + interest;
    } catch (Exception e)
    {
        return "Either principal value or rate value or time value is invalid";
    }
}
```

CFG

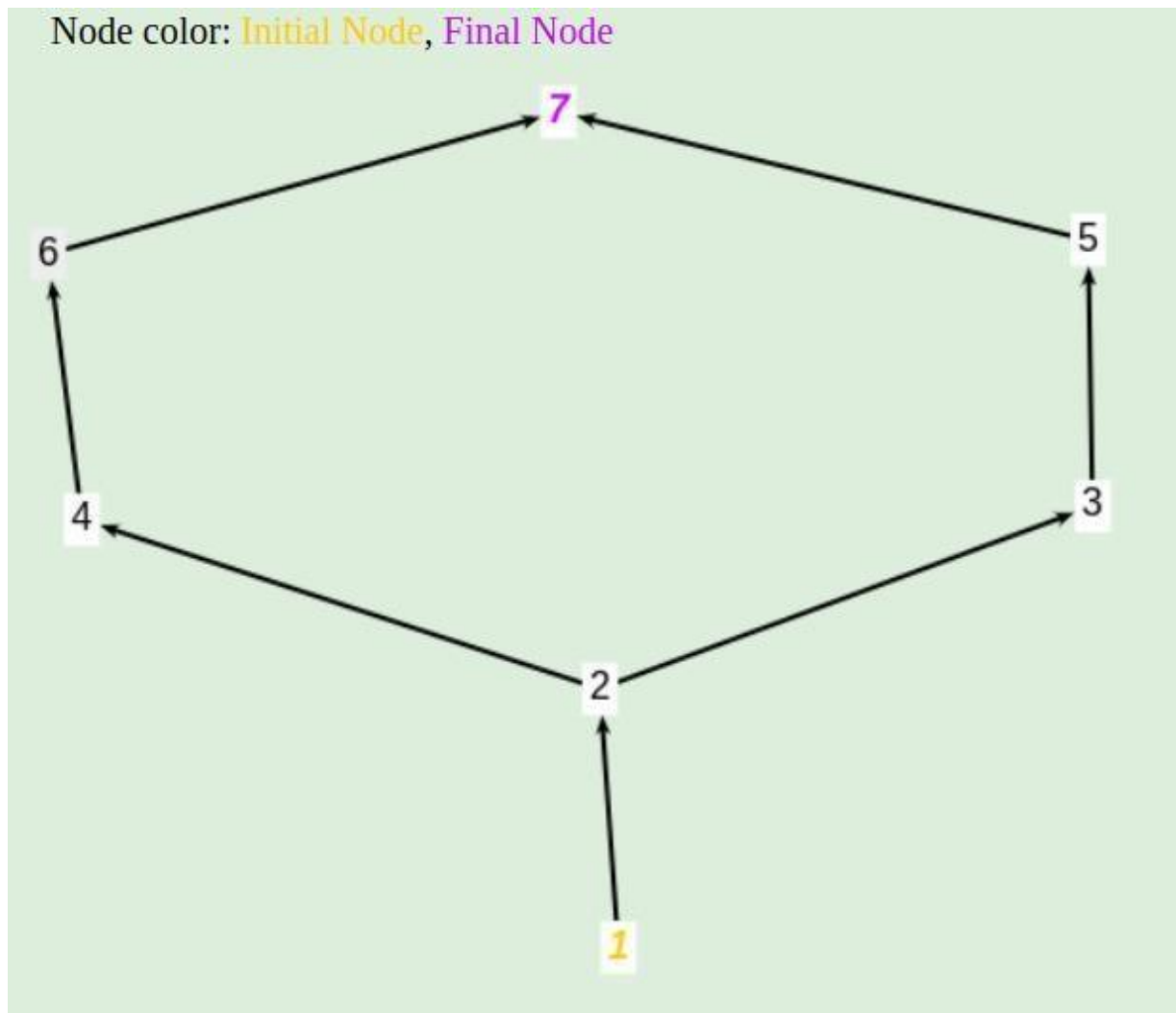


DFG



Graph, DU Pairs, Du Paths and All du path coverage

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|-----------|---------------------------|
| principal | [1,2] [1,2,4] |
| rate | [1,2] [1,2,4] |
| time | [1,2] [1,2,4] |
| exception | [3,5] |
| amount | No path or No path needed |
| interest | No path or No path needed |
| result | [4,6] |

DU Pairs for all variables are:

| Variable | DU Pairs |
|-----------|----------------|
| principal | [1,2] [1,4] |
| rate | [1,2] [1,4] |
| time | [1,2] [1,4] |
| exception | [3,5] |
| amount | [4,4] |
| interest | [4,4] |
| result | [4,6] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|-----------|---------------------------|
| principal | [1,2,4,6,7] |
| rate | [1,2,4,6,7] |
| time | [1,2,4,6,7] |
| exception | [1,2,3,5,7] |
| amount | No path or No path needed |
| interest | No path or No path needed |
| result | [1,2,4,6,7] |

Here we have two unique paths:-

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

Input:- principal=0, rate=10, time=20

Output:- Either the value of principal or the value of rate or the value of time is invalid.

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

Input:- principal=1000, rate=10, time=5

Output:- Amount is: 1610.5100000000004 Interest is:
610.5100000000004

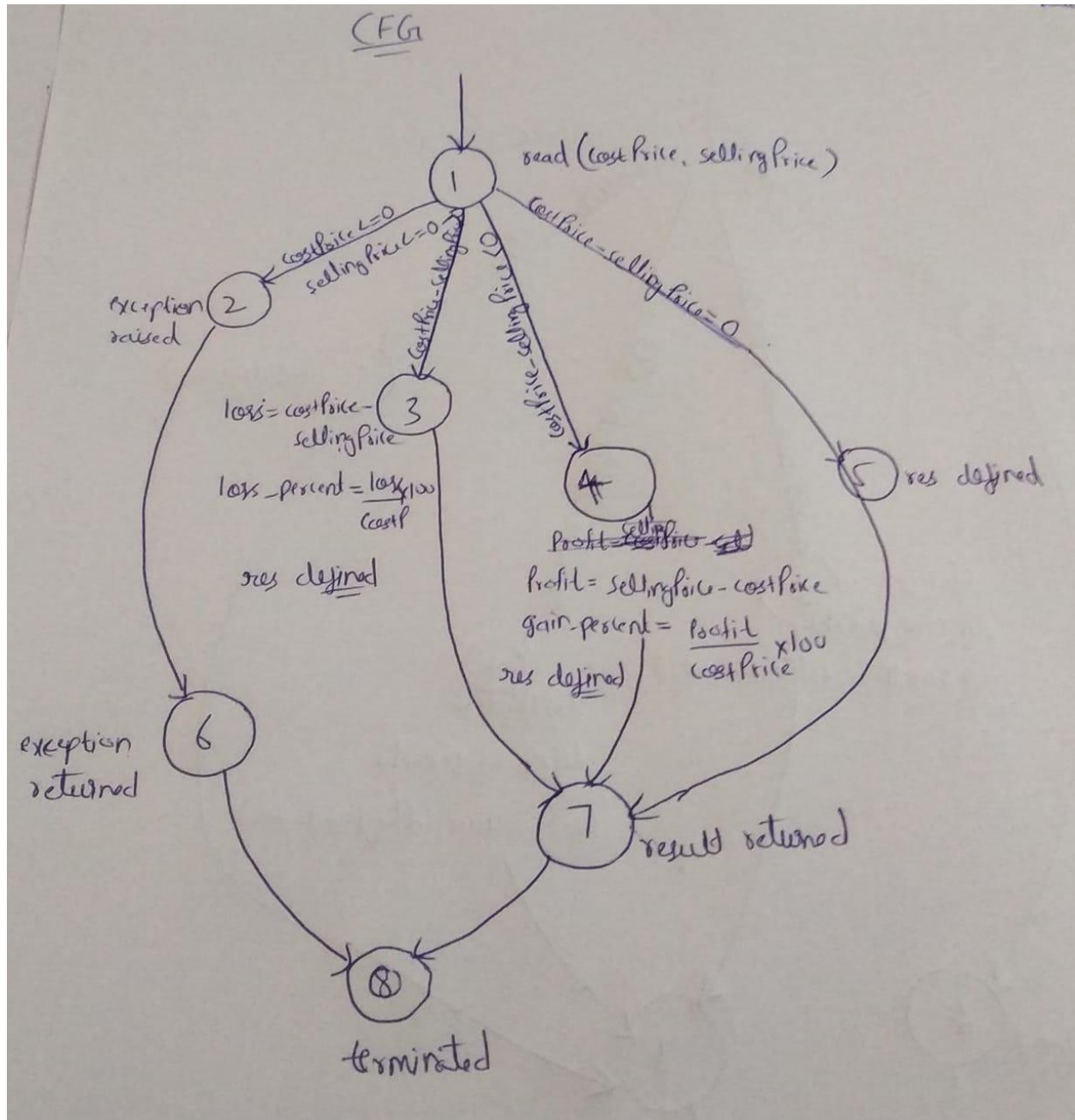
Result

| | |
|--|-------|
| ✓ compoundInterest()=>follows path [1,2,3,5,7] | 11 ms |
| ✓ compoundInterest()=>Follows path [1,2,4,6,7] | 50 ms |

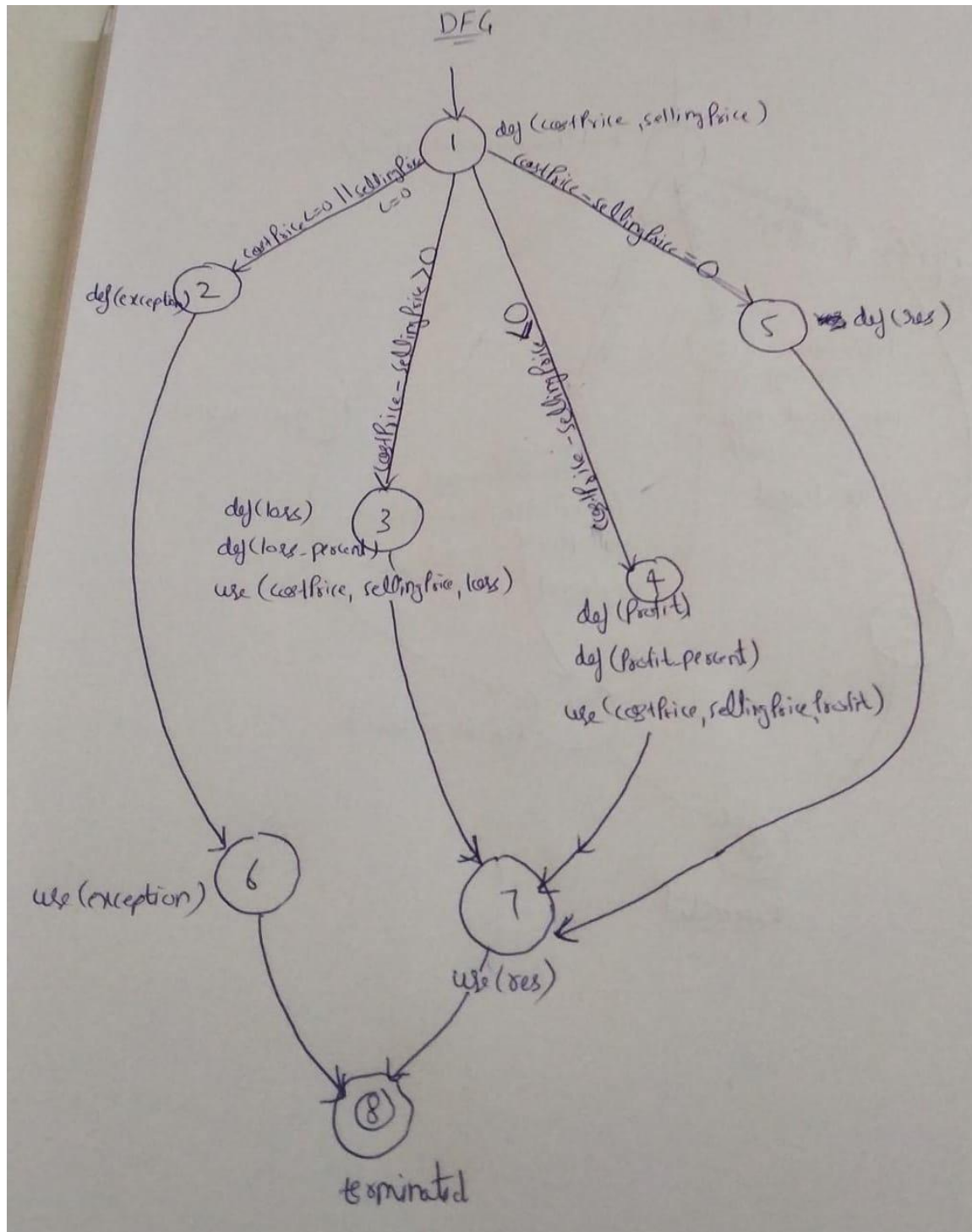
8. Find Profit and Loss:-

```
public String compareProfitLoss(double costPrice,double sellingPrice)
{
    String res;
    try {
        if (costPrice <= 0 || sellingPrice<=0) {
            throw new Exception();
        }
        else if(costPrice-sellingPrice>0) {
            double loss=costPrice-sellingPrice;
            double loss_percent=(loss/costPrice)*100;
            res="loss: " + loss + ", loss percent: " +loss_percent+"%";
        }
        else if(costPrice-sellingPrice<0) {
            double gain=costPrice-sellingPrice;
            double gain_percent=(1 * gain/costPrice)*100;
            res="profit: "+ gain +", profit percent: " +gain_percent+"%";
        }
        else {
            res="No profit, No loss";
        }
        return res;
    }
    catch (Exception e)
    {
        return "Either costPrice value or sellingPrice value is invalid";
    }
}
```

CFG



DFG

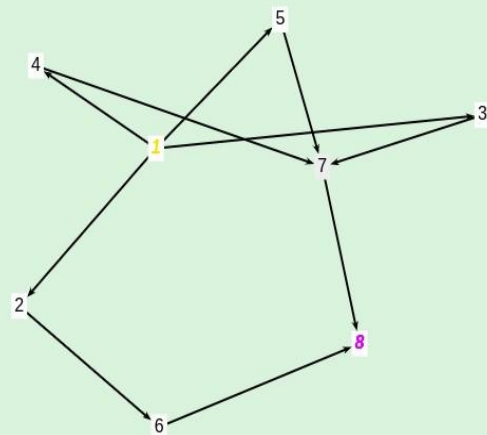


Graph, DU Pairs, Du Paths and All du path coverage

DU Pairs for all variables are:

| Variable | DU Pairs |
|---------------|----------|
| costPrice | [1,1] |
| | [1,3] |
| | [1,4] |
| sellingPrice | [1,1] |
| | [1,3] |
| | [1,4] |
| exception | [2,6] |
| loss | [3,3] |
| losspercent | [3,3] |
| profit | [4,4] |
| profitpercent | [4,4] |
| res | [3,7] |
| | [4,7] |
| | [5,7] |

Node color: Initial Node, Final Node



DU Paths for all variables are:

| Variable | DU Paths |
|---------------|---------------------------|
| costPrice | [1,3] |
| | [1,4] |
| sellingPrice | [1,4] |
| | [1,3] |
| exception | [2,6] |
| loss | No path or No path needed |
| losspercent | No path or No path needed |
| profit | No path or No path needed |
| profitpercent | No path or No path needed |
| res | [3,7] |
| | [4,7] |
| | [5,7] |

All DU Path Coverage for all variables are:

| Variable | All DU Path Coverage |
|---------------|-------------------------------------|
| costPrice | [1,3,7,8] [1,4,7,8] |
| sellingPrice | [1,4,7,8] [1,3,7,8] |
| exception | [1,2,6,8] |
| loss | No path or No path needed |
| losspercent | No path or No path needed |
| profit | No path or No path needed |
| profitpercent | No path or No path needed |
| res | [1,3,7,8] [1,4,7,8] [1,5,7,8] |

Here we have total 4 unique paths:-1.

[1,2,6,8]

Input:- costPrice=-10, sellingPrice=100

Output:- Either costPrice value or sellingPrice value is invalid.

2. [1,3,7,8]

Input:- costPrice=100, sellingPrice=80

Output:- loss: 20.0, loss percent: 20.0%

3. [1,4,7,8]

Input:- costPrice=100, sellingPrice=120

Output:- profit: 20.0, profit percent: 20.0%

4. [1,5,7,8]

Input:- costPrice=100, sellingPrice=100

Output:- No profit, No loss

Result

```

✓ compareProfitLoss()=>follows path [1,2,6,8]      1 ms
✓ compareProfitLoss()=>follows path [1,3,7,8]      96 ms
✓ compareProfitLoss()=>follows path [1,4,7,8]       7 ms
✓ compareProfitLoss()=>follows path [1,5,7,8]       2 ms

```

Contributions:-

1. Code, DFGs & CFGs, Du pairs, Du paths, and All Du path coverage TR and Test case design of `percentageChange()`, `findFirstNTermsAP()`, `simpleInterest()`, and `compareProfitLoss()` done by Gaurav Tilokani(MT2022045).
2. Code, DFGs & CFGs, Du pairs, Du paths, and All Du path coverage TR and Test case design of `fractionToPercentageConverter()`, `compoundInterest()`, `findFirstNTermsGP()`, and `getCountOfDays()` done by Jay Parekh(MT2022052).
