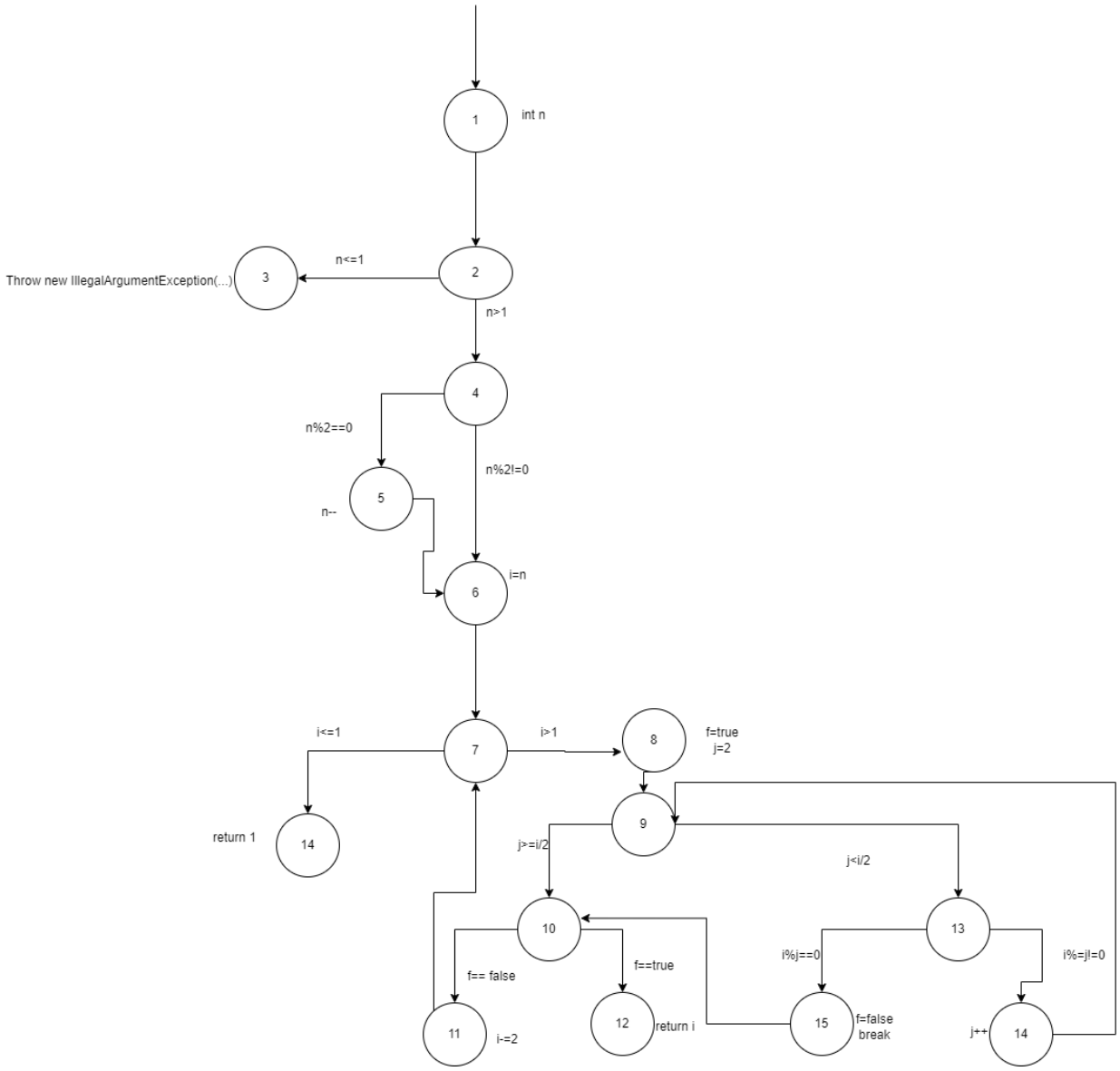
Втора лабораториска вежба по предметот СКИТ

```java
public static int closestLowerPrime(int n) {
    if(n <= 1)
        throw new IllegalArgumentException("The number has to be greater than 1.");

    if(n%2 == 0)
        n--;

    for(int i=n;i>1;i-=2) {
        boolean f=true;
        for(int j=2;j<i/2;j++) {
            if(i%j == 0) {
                f = false;
                break;
            }
        }

        if(f)
            return i;
    }

    return 1;
}
```

За ова најпрво треба да се нацрта графот за текот на контрола на програмата

(Control Flow Graph - CFG)



Во овој граф имаме еден почетен јазел јазелот 1, 3 терминални јазли 3,14,12.

Следно бараме usage и definition за секоја променлива

Јазел	Def	Use
1	{n}	/
6	{i}	{n}
5	{n}	{n}
8	{f,j}	/
11	{i}	{i}

12	/	{i}
14	{j}	{j}
15	{f}	/

Ребро	Use
(2,3)	{n}
(2,4)	{n}
(4,5)	{n}
(4,6)	{n}
(7,14)	{i}
(7,8)	{i}
(9,10)	{i,j}
(9,13)	{i,j}
(10,11)	{f}
(10,12)	{f}
(13,15)	{i,j}
(13,14)	{i,j}

Следно како Data Flow Criteria (DFC) ќе го користам All-du-paths coverage, како најчиесто користен критериум за покривање на тек на податоците

За таа цел, прво ќе ги најдеме сите du-pairs, односно парови од јазли каде што се дефинира променлива и јазли или ребра каде што се користи истата. Тоа ќе ни помогне за потоа да ги најдеме du-paths. Во следната табела може да ги видиме du-pairs за сите промелниви

Variable	DU Pairs
n	[1,5] [1,6] [5,5] [5,6] [1, (2, 3)] [1, (2, 4)] [1, (4, 5)] [1, (4, 6)]
i	[6,11] [6,12] [11,11] [11,12] [6, (7, 14)] [6, (7, 8)] [6, (9, 10)]

	[6, (9, 13)] [6, (13, 15)] [6, (13, 14)] [11, (7, 14)] [11, (7, 8)] [11, (9, 10)] [11, (9, 13)] [11, (13, 15)] [11, (13, 14)]
f	[8, (10, 12)] [8, (10, 11)] [15, (10, 12)] [15, (10, 11)]
j	[8,14] [14,14] [8, (9, 10)] [8, (9, 13)] [8, (13, 15)] [8, (13, 14)] [14, (9, 10)] [14, (9, 13)] [14, (13, 15)] [14, (13, 14)]

Variable	DU Paths
n	[1,2,3] [1,2,4,6] [5,6]
i	[6,7,14] [6,7,8,9,13,15] [6,7,8,9,13,14] [6,7,8,9,10,12] [11,7,8,9,10,12] [11,7,14] [11,7,8,9,13,15] [11,7,8,9,13,14]
f	[8,9,10,12] [8,9,10,11] [15,10,11] [15,10,12]

j	[8,9,10] [8,9,13,15] [14,9,10] [14,9,13,15]

За променливата i сите патеки што ги вклучуваат јазлите 6 и 11 заедно не се def-clear па тие не ги земаме во предвид.

За променливата j сите патеки што ги вклучуваат јазлите 8 и 14 заедно не се def-clear па тие не ги земаме во предвид

Следната листа на патеки е дополнителнопрочистена листа од патеките кои не се јавуваат како подпатеки.

1. [1,2,3]
2. [1,2,4,6]
3. [5,6]
4. [6,7,14]
5. [6,7,8,9,13,15]
6. [6,7,8,9,13,14]
7. [6,7,8,9,10,11]
8. [6,7,8,9,10,12]
9. [11,7,8,9,10,12]
10. [11,7,14]
11. [11,7,8,9,13,15]
12. [11,7,8,9,13,14]
13. [8,9,10,11]
14. [15,10,11]
15. [15,10,12]
16. [14,9,10]
17. [14,9,13,15]

[1,2,3] е тест патека сама по себе

[1,2,4,5,6,7,14] ни ги покрива тест патеките 3 и 4

[1,2,4,6,7,8,9,10,11,13,15] ни ги покрива 2,5 и 13

[1,2,4,6,7,8,9,10,11,7,8,9,13,15,10,12] ни ги покрива 6,14,16 и 8

[6,7,8,9,10,11] ни го покрива 7

[1,2,4,6,7,8,9,10,11,7,8,9,13,15,9,10,12] ни ги покрива 11 и 15

[1,2,4,6,7,8,9,10,11,7,8,9,10,12] ни го покрива 9

[1,2,4,6,7,8,9,10,11,7,8,9,13,15] ни го покрива 10

[1,2,4,6,7,8,9,10,11,7,8,9,13,14]

Test Cases

1. Патека [1,2,3]

   n=1

   Output: Invalid Output Exception

2. Патека:[1,2,4,5,6,7,14]
   n=2
   Output=1
3. Патека :[1,2,4,6,7,8,9,10,11,13,15]
   Infeasible Test Case
   Output:NaN
4. Патека: [1,2,4,6,7,8,9,13,14,10,11,7,14]
   N=2
   Output 1
5. Патека: [6,7,8,9,10,11]
   Infeasible Test Case
   Output:NaN
6. Патека: [1,2,4,6,7,8,9,10,11,7,8,9,13,15,9,10,12]
   Infeasible Test Case
   Output:NaN
7. Патека: [1,2,4,6,7,8,9,10,11,7,8,9,10,12]
   Infeasible Test Case
   Output:NaN
8. Патека: [1,2,4,6,7,8,9,10,11,7,8,9,13,15]
   Infeasible Test Case
   Output:NaN
9. Патека: [1,2,4,6,7,8,9,10,11,7,8,9,13,15]

Infeasible Test Case
Output:NaN