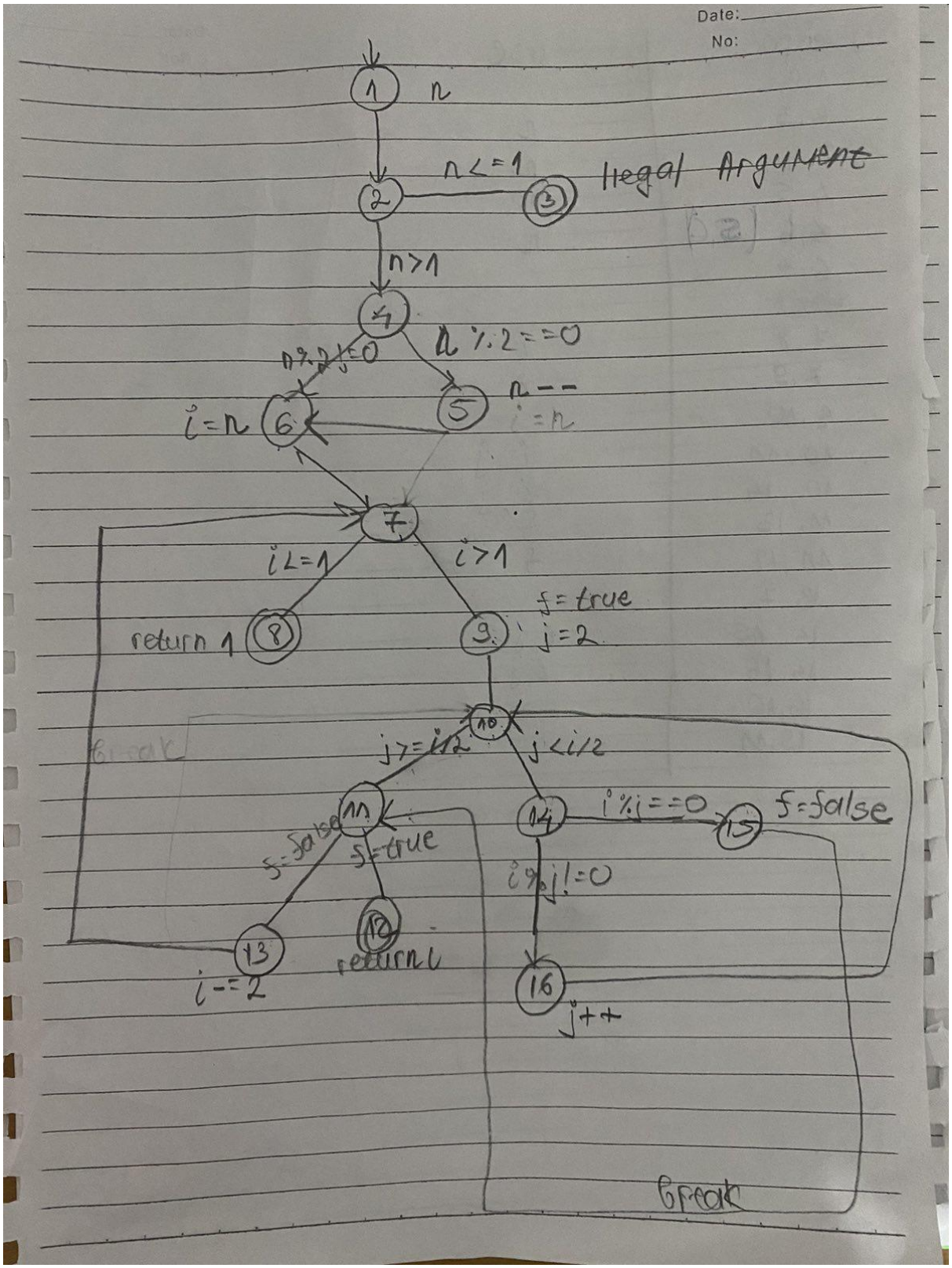
Христина Секулоска

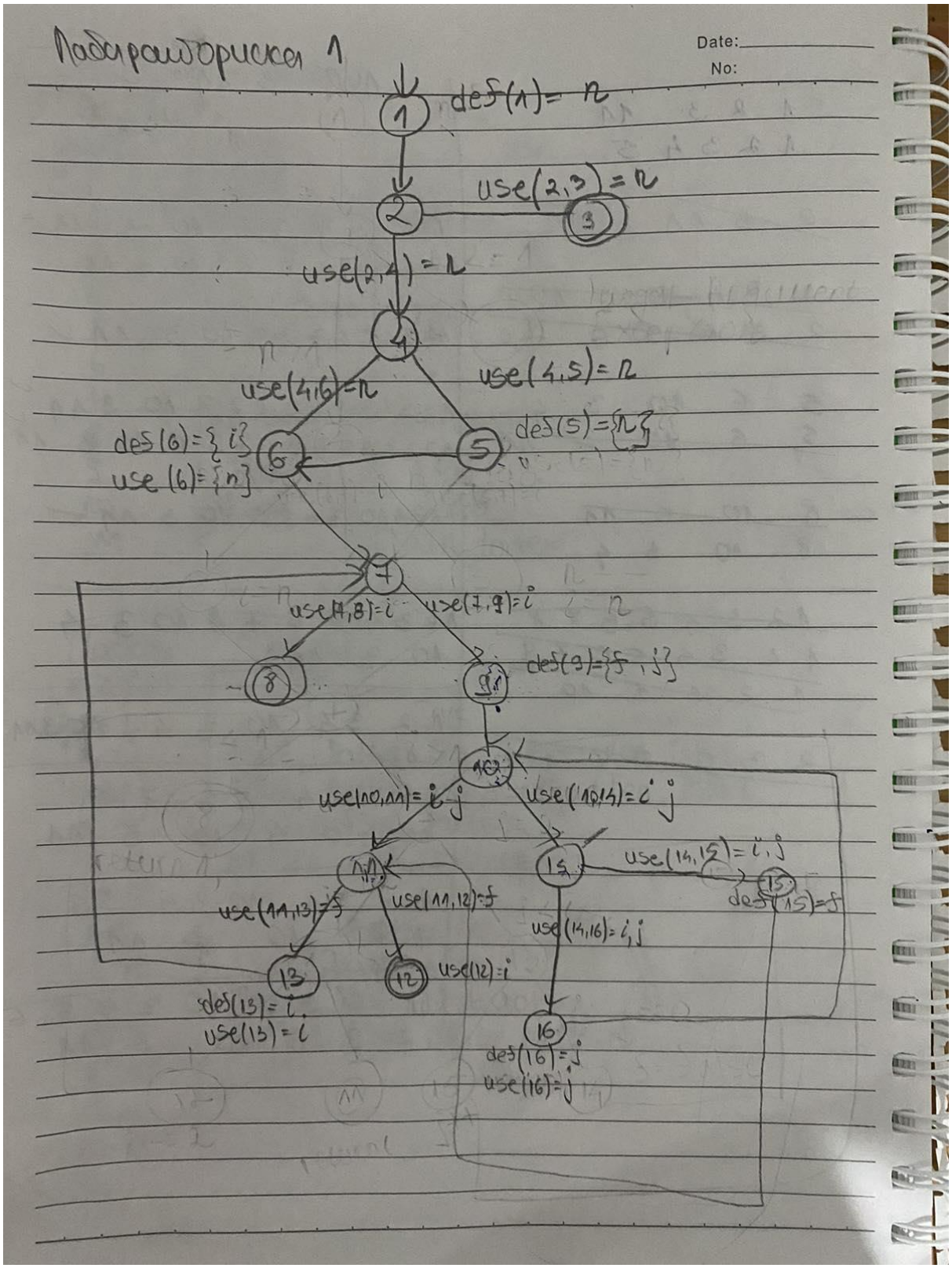
211236

```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;
}
```

На цртежот во прилог е нацртан графот за текот на контрола на програмата (Control Flow Graph - CFG).



Граф на тек на податоците.



Табела за defs и usages за јазли

Јазел	def	use
1	n	
2		
3		
4		
5	n	
6	i	n
7		
8		
9	f, i	
10		
11		
12		i
13	i	i
14		
15	f	
16	j	j

Табела usages на ребра

Ребро	use
1,2	
2,3	n
2,4	n
4,5	n
4,6	n
5,6	
6,7	
7,8	i
7.9	i
9,10	
10,11	i,j
10,14	i,j
11,13	f
11,12	f
13,7	
14,15	i,j
14,16	i,j
16,10	

15,11	

All-du-paths coverage е најчесто користен критериум за покривање на тек на податоците. Најпрво за таа цел треба да се најдат сите du-paths. DU-path е патека која почнува во јазел којшто е дефинирана некоја променлива, и завршува во јазел или ребро каде што се користи истата таа променлива, а е def clear, односно внатре во патеката не постојат други јазли во коишто повторно се дефинира таа променлива. Во прилог е табелата со сите DU Paths.

Variable	DU Paths
n	[1,2,4] [1,2,3] [1,2,4,6] [1,2,4,5] [5,6]
i	[6,7,9] [6,7,8] [6,7,9,10,14] [6,7,9,10,11] [6,7,9,10,14,15] [6,7,9,10,14,16] [6,7,9,10,11,13] [6,7,9,10,11,12] [6,7,9,10,14,15,11,13] [6,7,9,10,14,15,11,12] [13,7,9] [13,7,8] [13,7,9,10,14] [13,7,9,10,11] [13,7,9,10,14,15] [13,7,9,10,14,16] [13,7,9,10,11,13] [13,7,9,10,11,12] [13,7,9,10,14,15,11,12] [13,7,9,10,14,15,11,13]
f	[9,10,11,12] [9,10,11,13] [15,11,13] [15,11,12]
j	[9,10,14] [9,10,11]

	[9,10,14,16] [9,10,14,15] [16,10,14] [16,10,11] [16,10,14,15] [16,10,14,16]

Согласно сите DU Paths ги означив со црвено тие што не се def clear.
Така што, овие патеки нема да ги земеме во предвид.
Имено, можеме да приметиме дека некои од патеките се под-патеки (префикс) на други, а некои дури и целосно се поклопуваат. Затоа ваквите патеки може и да ги изоставиме и да ги земеме само оние уникатни што ги содржат сите. После таквото филтрирање ги добиваме овие du-paths:

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

Според овие патеки, ги добиваме следните тест патеки:

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

[1,2,4,6,7,9,10,14,16,10,14,15,11,12]
[1,2,4,6,7,9,10,14,16,10,14,16,10,11,12]

● Тест 1
Патека: [1,2,4,6,7,8]
n = 1
очекуван излез: return 1
● Тест 2
Патека: [1,2,4,5,6,7,8]
n = 2
очекуван излез: return 1
● Тест 3
Патека: [1,2,4,6,7,9,10,11,12]
n = 3
i=1.5
очекуван излез: return 3
● Тест 4
Патека: [1,2,4,6,7,9,10,14,15,11,12]
n = 9
i=4.5
очекуван излез: return 1
● Тест 5
Патека: [1,2,4,6,7,9,10,14,16,10,11,12]
n = 7
i=3.5
очекуван излез: return 7
● Тест 6
Патека: [1,2,4,6,7,9,10,14,16,10,11,12]
n = 7
i=3.5
очекуван излез: return 7
● Тест 7
Патека: [1,2,4,6,7,9,10,11,13,7,8]
n = 3
i=1
очекуван излез: return 3
● Тест 8
Патека: [1,2,4,6,7,9,10,14,15,11,13,7,8]
n =12
i=4
очекуван излез: return 1
● Тест 9
Патека: [1,2,4,6,7,9,10,11,13,7,9,10,11,12]
n =3

i=1.5
очекуван излез: 3