Homework 5 - Bearbeitung

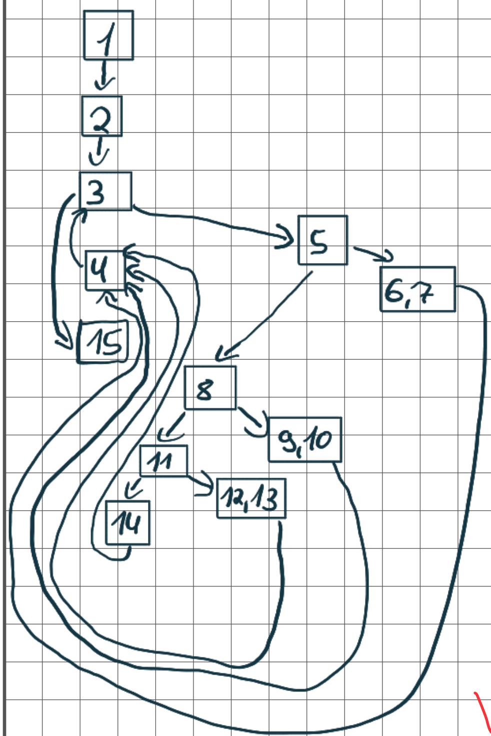
Nr 1

1. Bemerkungen wie „Basic Programming Knowledge :)" könnten als abwertend empfunden werden, stattdessen sollte man lieber konstruktiv und respektvoll formulieren.
2. Man könnte spezielle positive Aspekte am Code herausheben, um respektvoller zu wirken.
3. Man könnte konkrete Verbesserungsvorschläge machen statt zu schreiben „Just try to make it simpler"
4. Testfälle könnten weiter konkretisiert werden. Man könnte Vorschläge über fehlende TestCases (bzw. Landcases) machen, statt einfach zu schreiben es könnten mehr sein
5. Bei der Schleife könnte man Vorschläge machen, wie man diese schneller machen könnte statt zu schreiben „I recommend trying to make it faster"

Nr. 2) Boundary Testing fehlt bei den markierten Klassen. -0,5P

Test Cases	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9
totalS ≤ 0	X								
totalS > 0		X	X	X	max	X	X	max	min
groupSize ≤ 0		X							
groupSize > 0	X		X	X	X	max	min	max	X
availableGr ≤ 0			X						
availableGr > 0	X	X		X	min	X	max	max	X
Result = 0	X								
Result > 0				X	X	X	X	X	X
exception		X	X						
input totalS	0	2	2	10	maxInt	10	20	maxInt	1
input groupSize	5	0	3	10	2	maxInt	1	maxInt	2
input available	5	3	0	5	1	3	maxInt	maxInt	3
expected output	0	Exception	Exception	10	2	10	20	maxInt	1

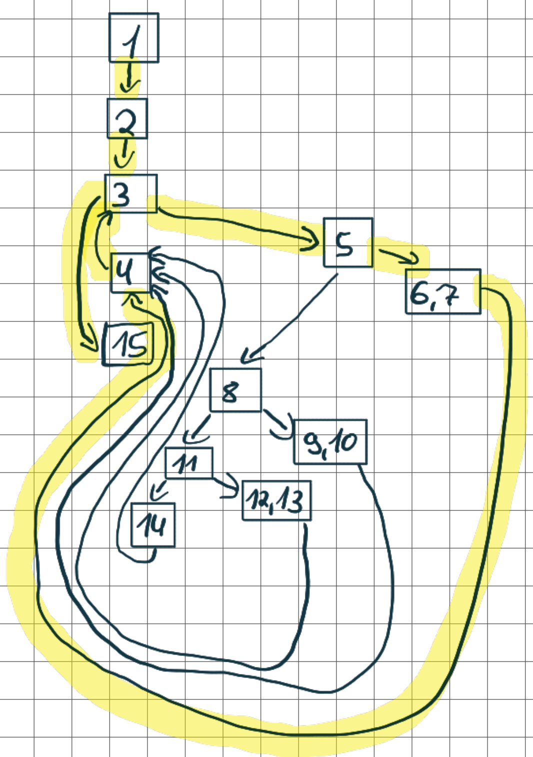
Nr 3
Control flow graph:



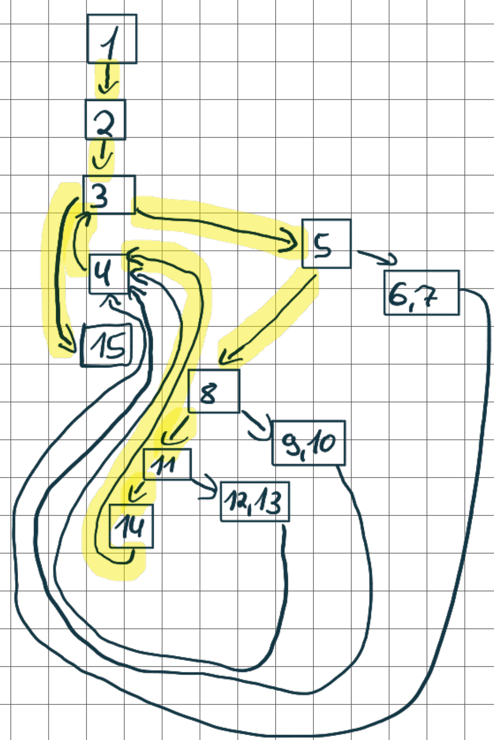
Die Nummer im control flow Graph entsprechen den Zeilennr.:

```
List<Student> assignedStudents = new ArrayList<>(); //1
          //2       //3                  //4
for(int i = 0; i < students.size(); i++) {
    if(students.get(i) == null
            || student.get(i).getID() == null) { // 5
        System.out.println("Invalid student or student ID"); //6
        continue; // 7
    }
    if(assignedStudents.contains(student)) { // 8
        System.out.println("Student already assigned"); // 9
        continue; // 10
    }
    if(assignedStudents.size() >= group.getCapacity()) { // 11
        System.out.println("Group is full"); //12
        continue; //13
    }

    // All checks passed, add student to group
    assignedStudents.add(student); //14
}

return assignedStudents; //15
}
```

Test 1 durchläuft folgenden Weg im Graph:



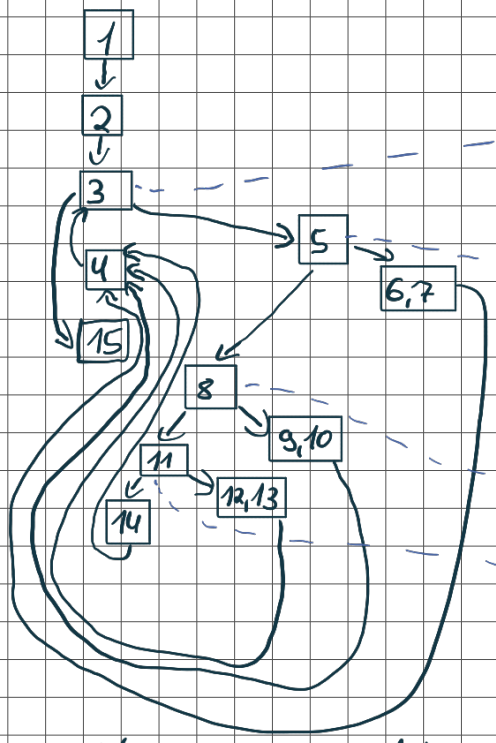
Test2 durchläuft folgenden Weg im Graphen:



$\Rightarrow$ Statement Coverage = $\frac{\text{lines covered}}{\text{lines total}} = \frac{11}{15} \approx 73{,}33\,\%$

$\Rightarrow$ Branch Coverage = $\frac{\text{decision outcomes covered}}{\text{decision outcomes total}} = \frac{6}{8} = 75\,\%$

Folgende Condition gibt es (die gelb markierten sind durch die Testfälle erfüllt):



- (i < students.size())
- ¬(i < students.size())
- (students.get(i) == null)
- ¬(students.get(i) == null)
- (students.get(i).getID() == null)
- ¬(students.get(i).getID() == null)
- (assignedStudents.contains(student))
- ¬(assignedStudents.contains(student))
- (assignedStudents.size() ≥ group.getCapacity())
- ¬(assignedStudents.size() ≥ group.getCapacity())

$\Rightarrow$ Condition coverage: $\frac{\text{condition outcomes covered}}{\text{condition outcomes total}} = \frac{7}{10} = 70\,\%$ ✓

Path coverage: $\frac{\text{paths covered}}{\text{paths total}}$ = Kann nicht berechnet werden, da die Schleife theoretisch unendlich oft durchlaufen werden kann, da die Liste students theoretisch unendlich groß werden kann

↳ Richtig. Was können wir in dem Fall machen?

z.B. das Loop Adequacy Criterion anwenden. -1p