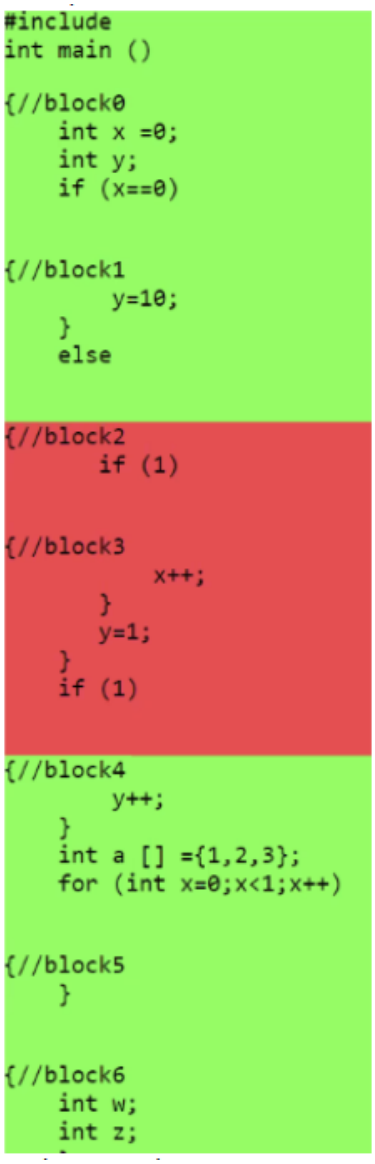
Compiler Project

Here is the description of the project:

Build a dynamic code analyzer for Java Language to generate statement and branches code coverage reports. Kindly refer to this stackoverflow link to know the difference between statement and branch coverage.

To warm you up, check the expected output of the project that you are expected to deliver at the end of the project:



Were the green lines show the executed commands, and the red lines show the commands which are not executed when the program ran.

The delivery of the project will consist of 3 phases, where a phase is delivered each week

Delivery 1 (18/3 —> 24/3):

- Each team has to create a github repository and add the team members to it
- Each team has to use ANTLR Java grammar, like the one provided at this link
- Each team has to test the grammar on a simple Java program to show the parse tree using ANTLR plugin with Intelli-J (A screenshot of parse tree).
- Each team has to show the starting rule of the grammar.
- Write a Java program based on Antlr that takes a java file as an input and outputs a modified intermediate java file (injected code) as demonstrated in the section where:

  you add comment in each block of this code indicates the number of this block , it should look like this :

  // block number 3

Delivery 2 (25/3 —> 31/3):

- Each team has to write a Java program based on Antlr that takes a java file as an input and outputs a modified intermediate java file (injected code) as demonstrated in the section where :

  when you run The modified intermediate java file (generated from the previous step) to know which blocks of the code are visited (A file has to be generated after the program run that shows which blocks are visited).

  the file should look like this

  block number 1 is visited

  block number 3 is visited

  …

Delivery 3 (15/4 —> 21/4): “you have two weeks for this delivery”

- Any missing parts of delivery 1 or 2 have to be finalized
- Use the output from delivery 2 to generate an HTML with highlighted red/green lines (as shown in the attached picture above)
    - Red lines are highlighted for not visited lines
    - Green lines are highlighted for visited lines
- Make sure that your pipeline does not include any manual effort
- This step means that your code is expected to take an input java file path and it does everything else on its own, like generating edited java files, the output visited blocks from delivery 2, and the generated HTML code file with green and red lines
- implement branch coverage
    - For any branches (if/else/for/while) if the boolean expression has more than one condition, like a || b, and the first condition always evaluate to true, this means that the second condition b never executed. In this case, this line must have an orange color in the output HTML file. If in the previous example, both conditions a and b are evaluated, then its output background color in the HTML file is green. If neither conditions a or b evaluate to true, then the boolean expression has a red color in the output HTML file.
- At least 3 java examples have to be uploaded to your github repo that shows difficult scenarios that your code cover (like function calls, nested conditions, if, else if, else, while, for)
- Each input java file must be uploaded to the Git repo with all its deliverables in deliveries 1, 2, and 3