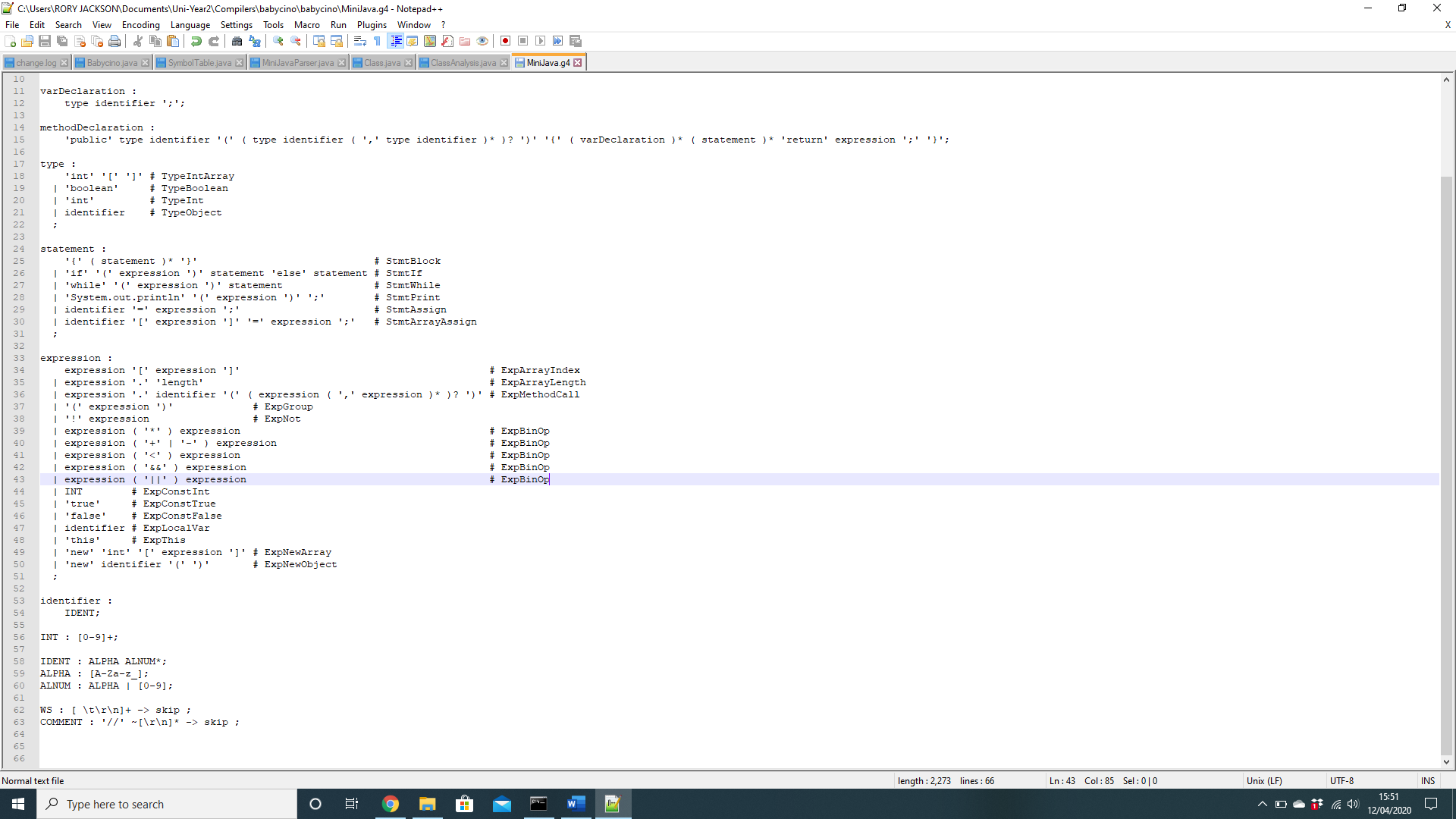
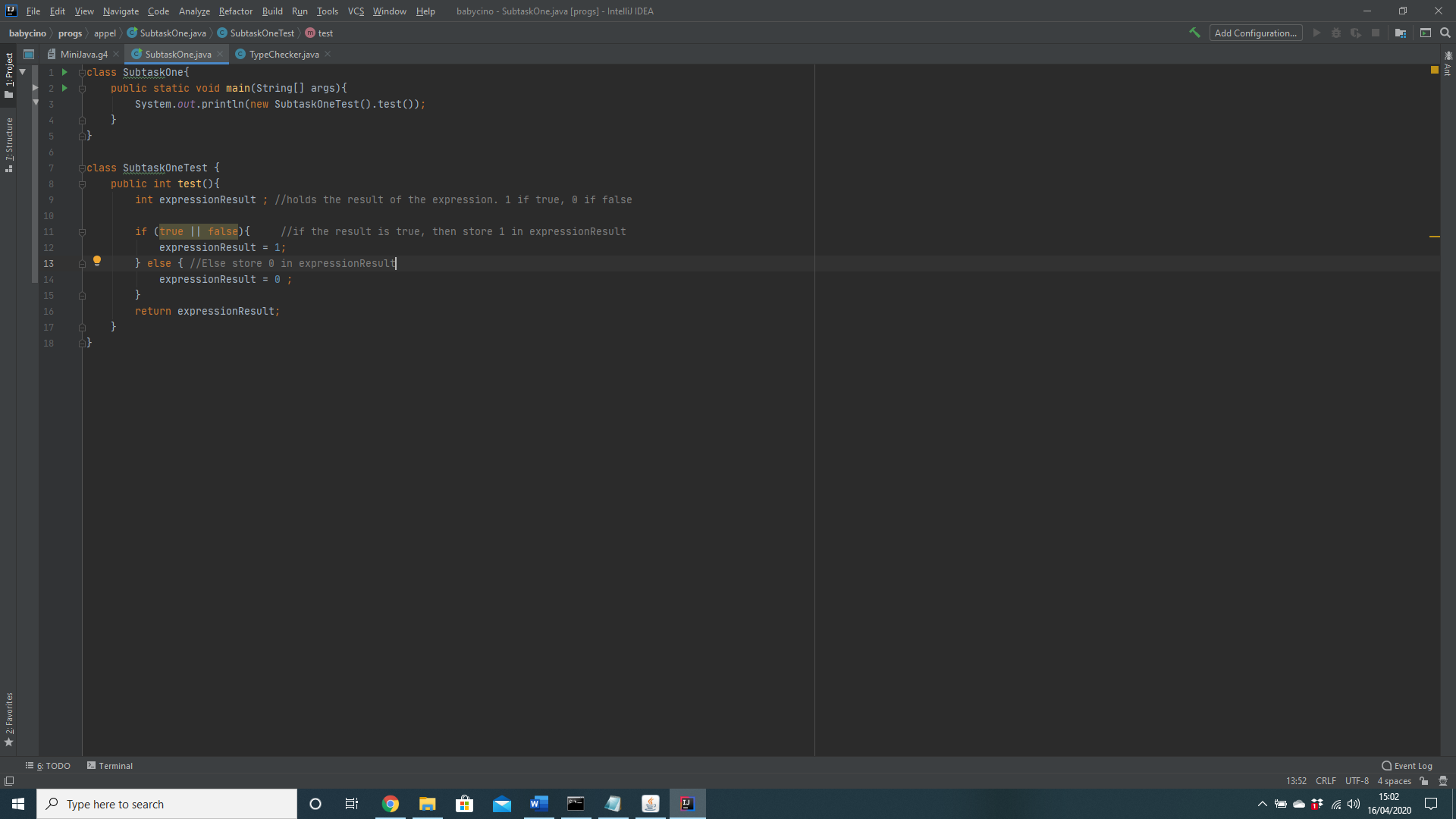
Module Code: CS2CO16  
Assignment Report Title: Adding Boolean OR to a compiler  
Student Number: 27002688  
Date of completion:  
Actual time spent on assignment:  
Assignment evaluation (3 key points):

**Subtask 1: Parsing (20%)**

1. Screenshot



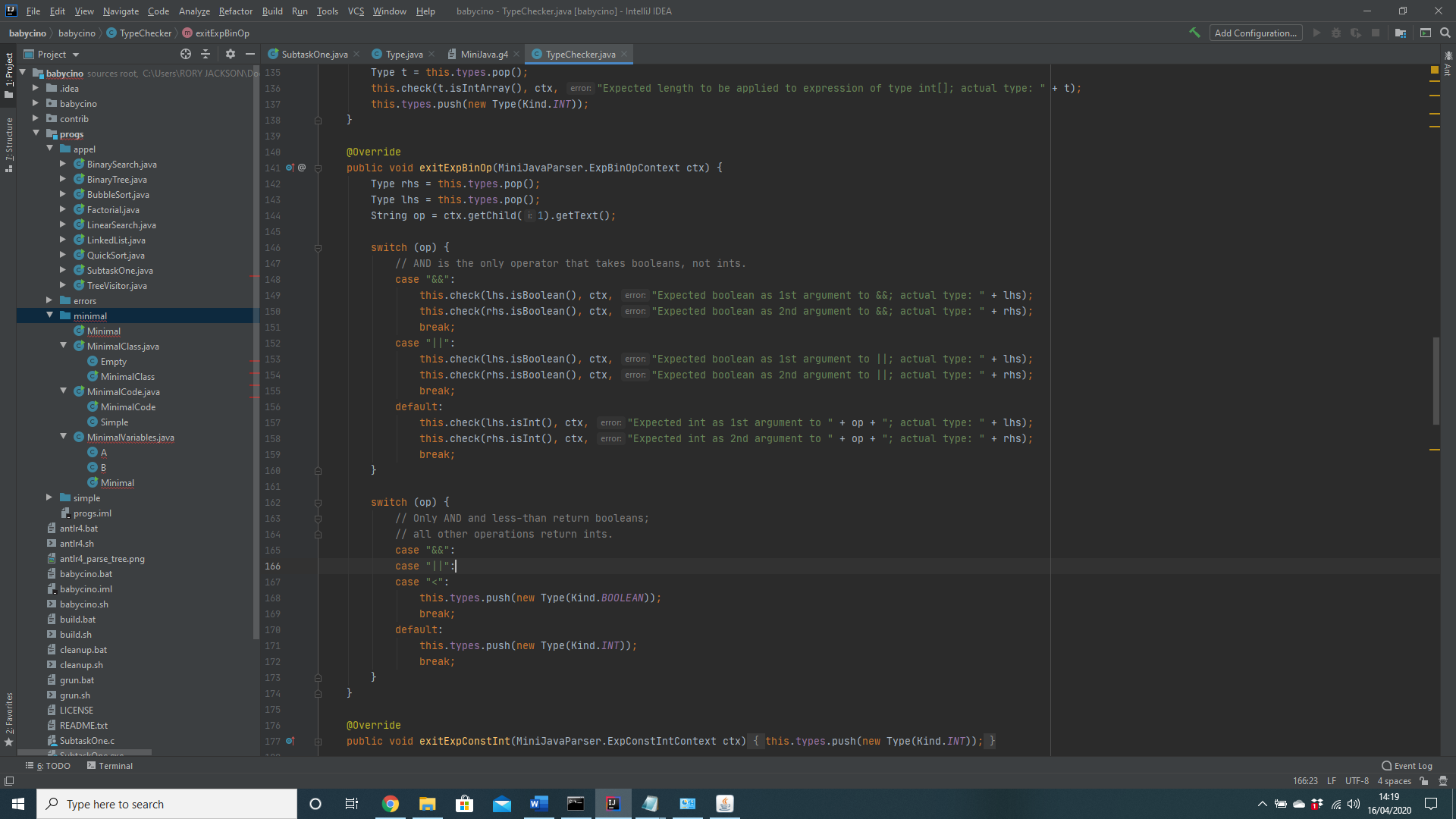
1. In order to modify babycino to parse || expressions, I had to modify the file MiniJava.g4 in order to accept || expressions.
2. Below is the code that contains a single || expression along with the parse tree for this program.



A close up of a logo

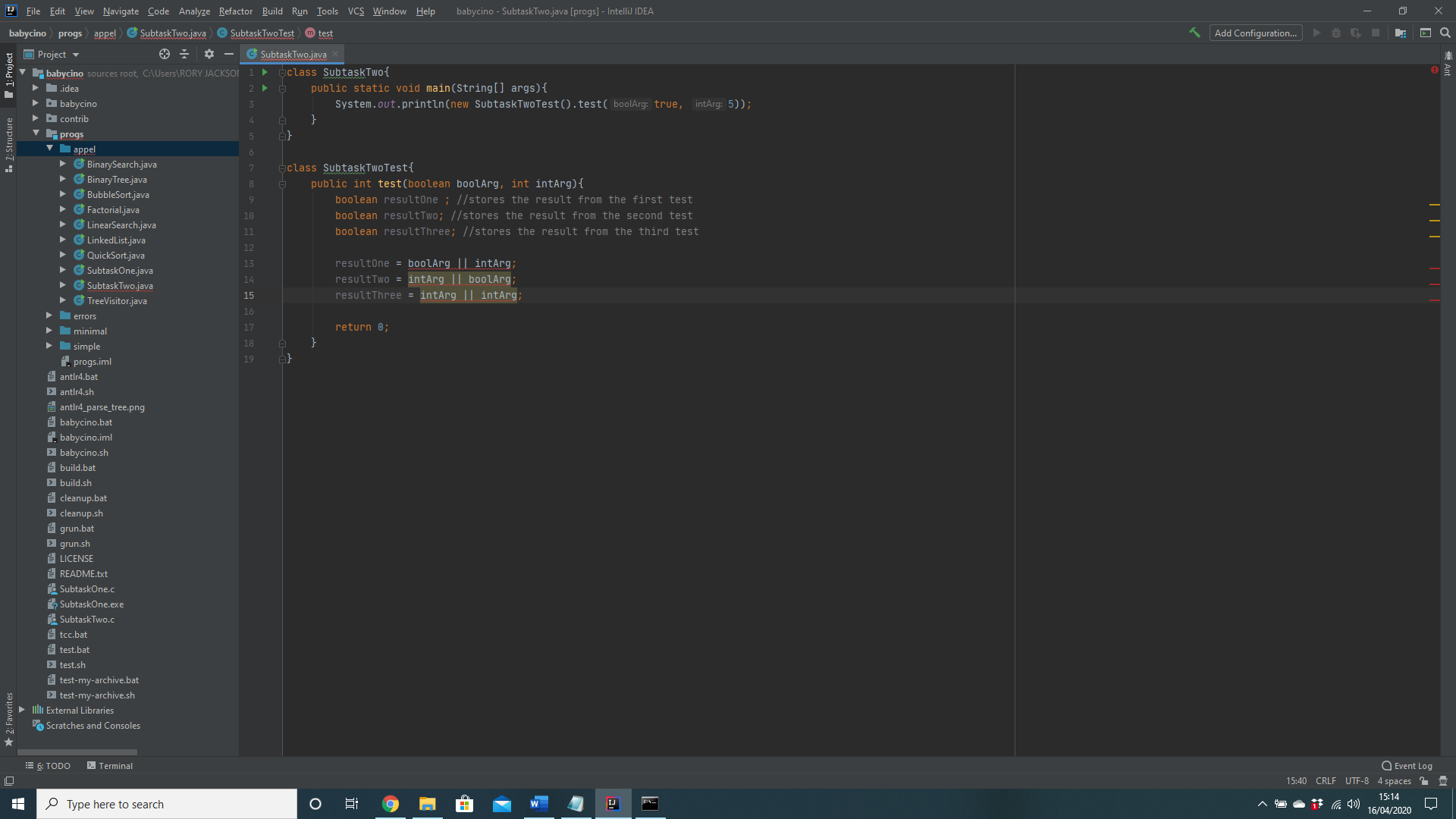
Description automatically generated

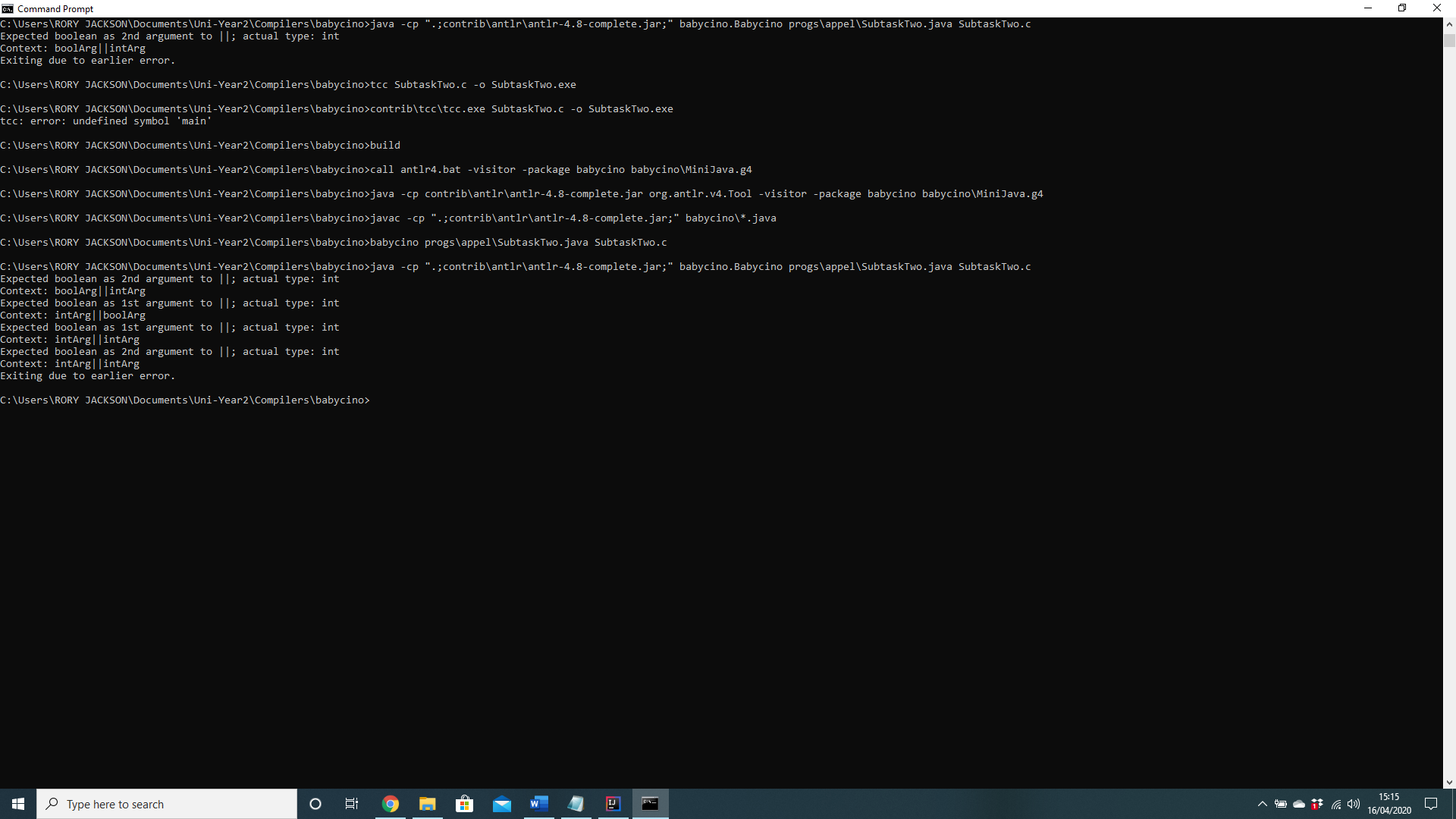
**Subtask 2: Semantic Analysis**

1. 

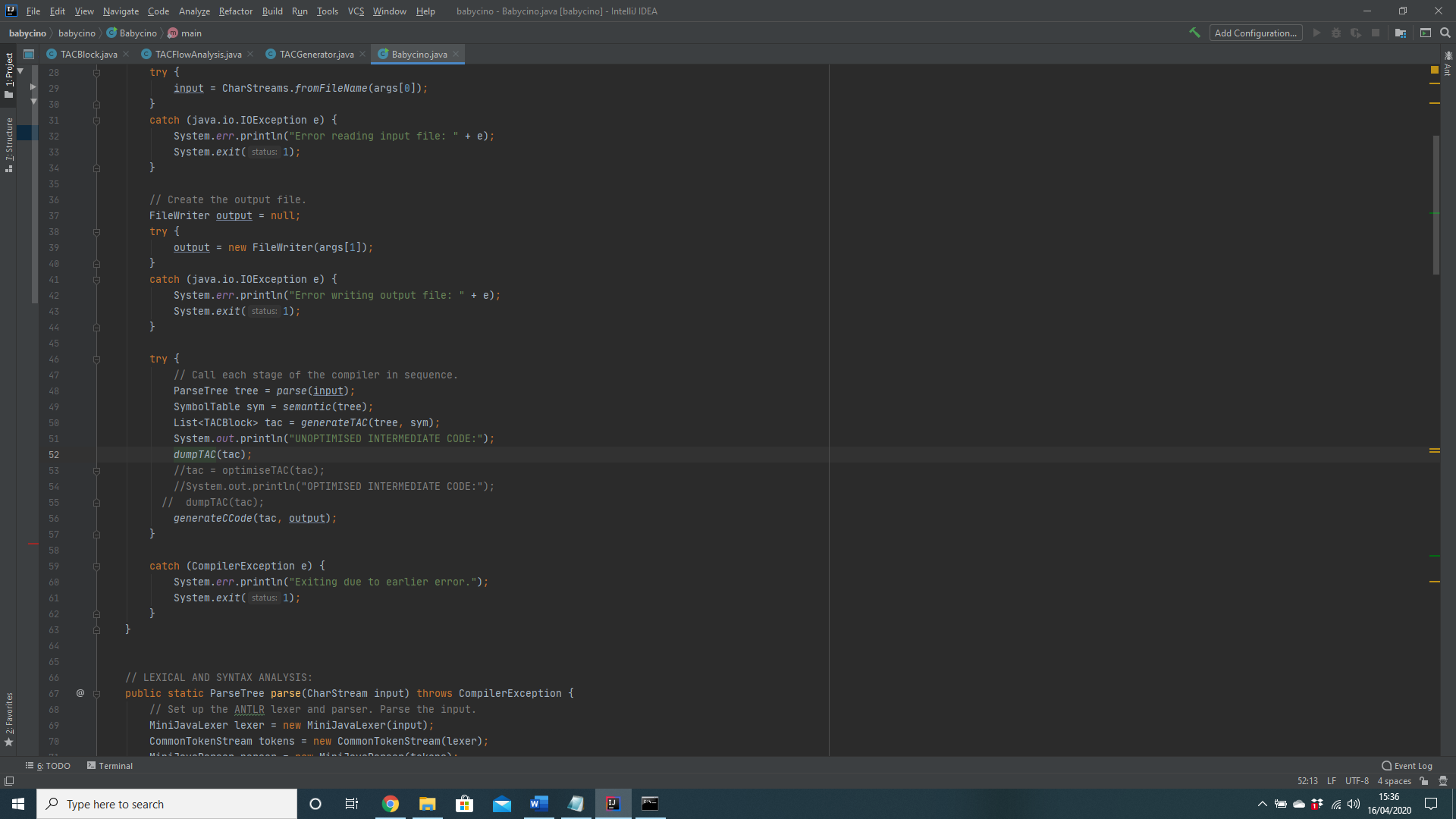
2. For babycino to reject ill-formed || expressions, the || had to be included in the type checker. As || is a Boolean operator, babycino needed to check the type of the input. Therefore, the TypeChecker.java file was modified in order to make babycino only accept Boolean inputs for ||.

3. Below is the code for the MiniJava program to reject ill-formed || expressions. The program tests the || against an int and a Boolean parameter in three separate tests.



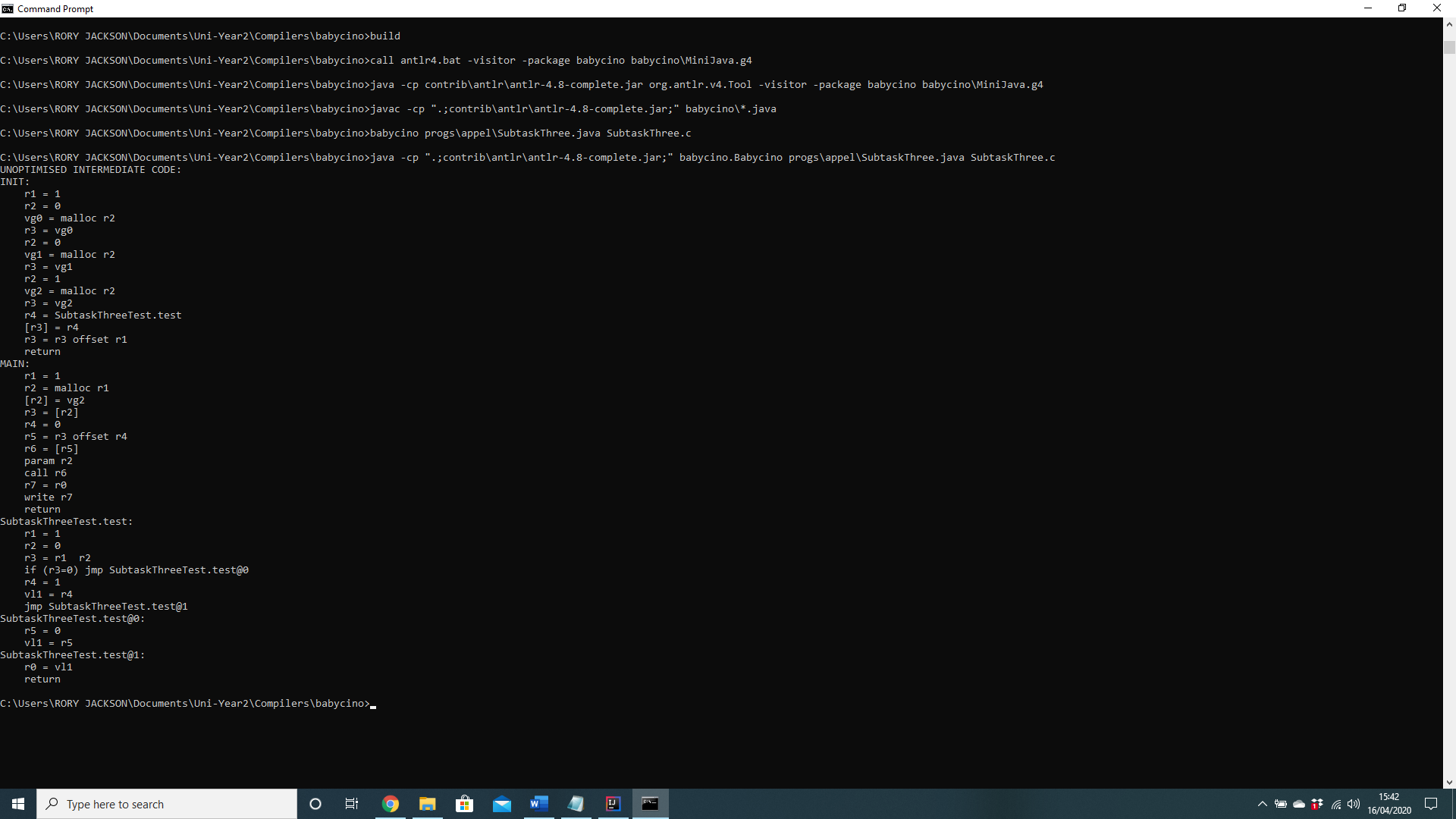


**Subtask 3: Code Generation**

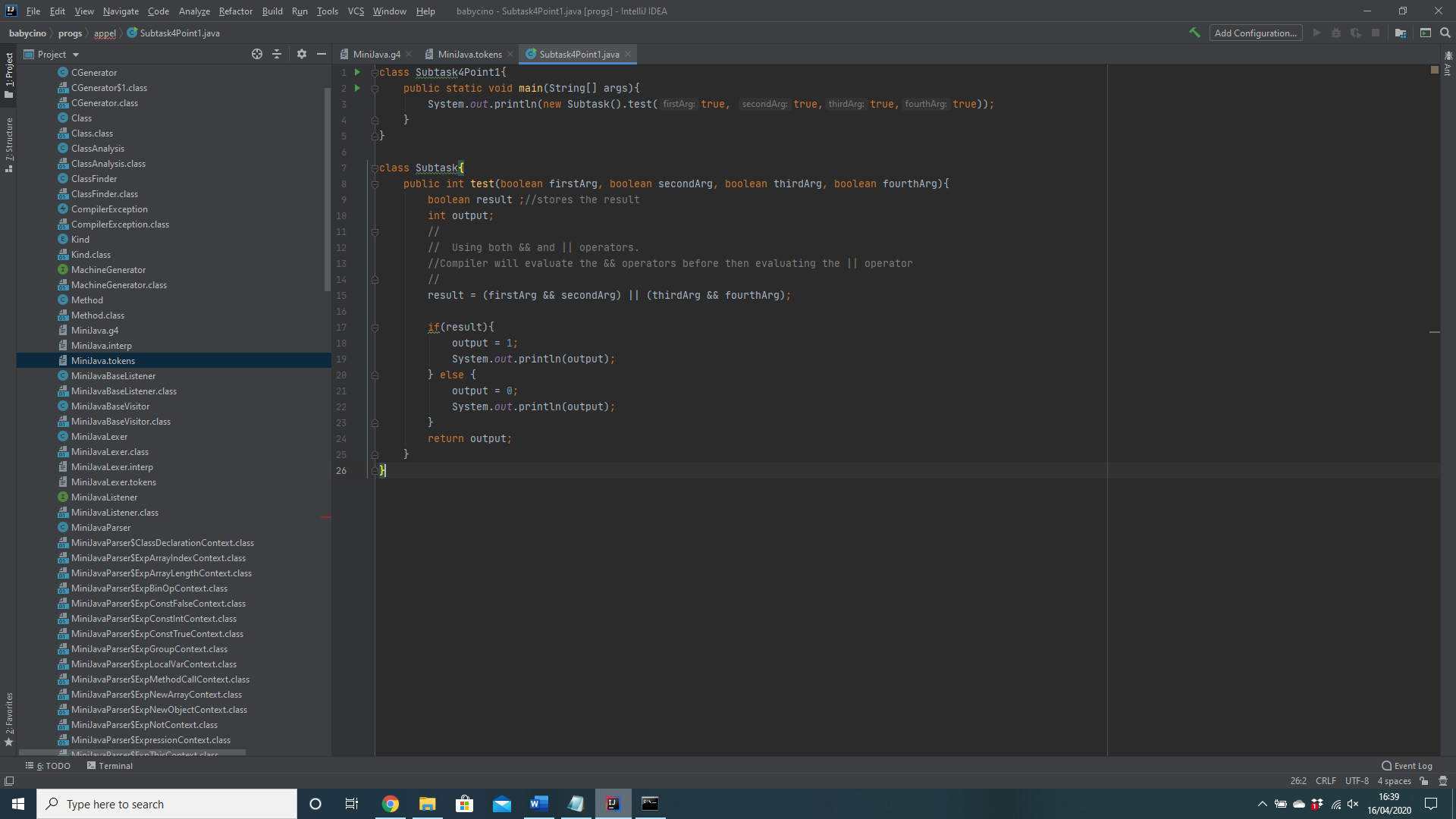
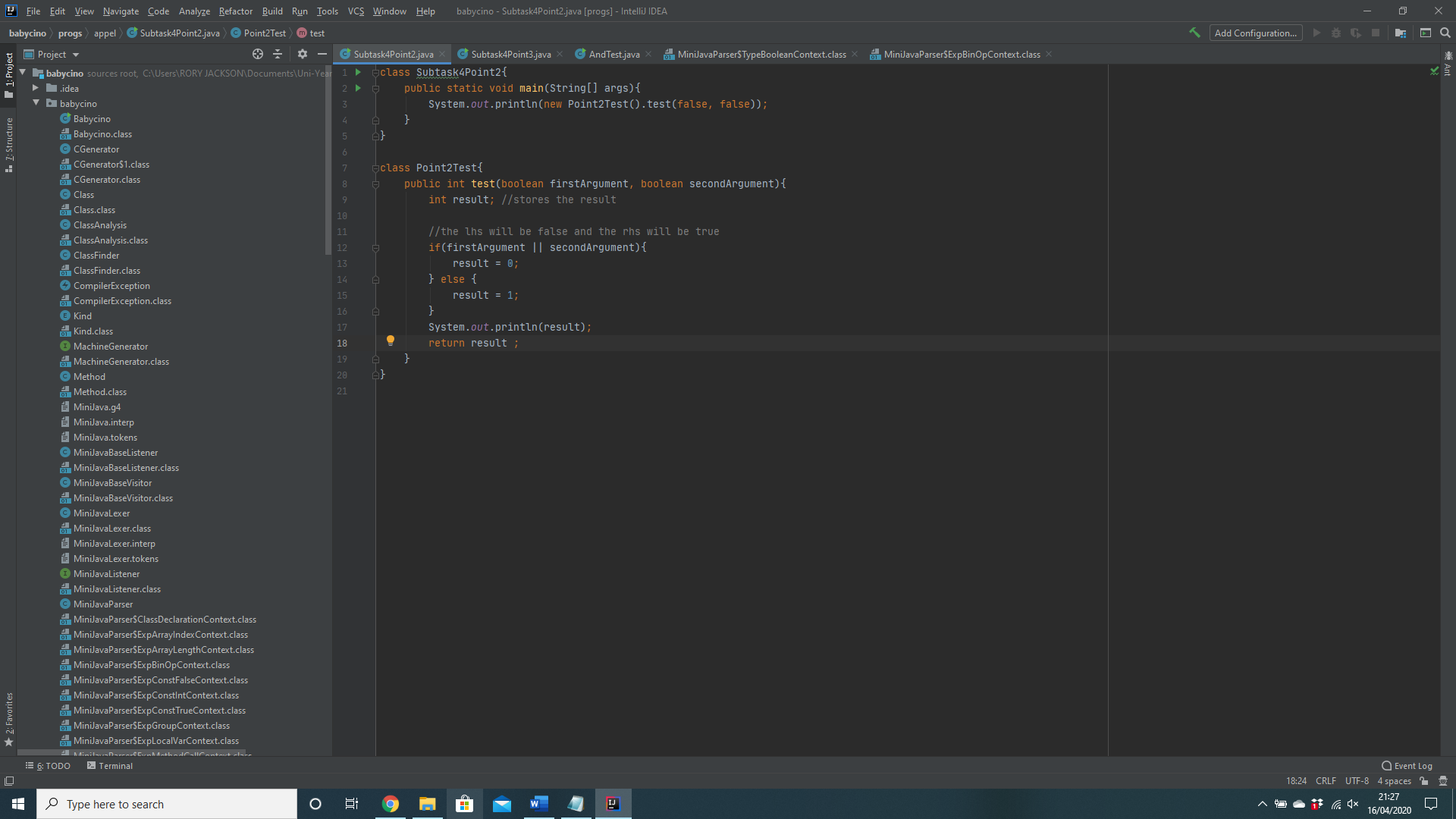
1. 

2. In order to generate the unoptimized intermediate code, the file *babycino.java* was edited. The code that contained the “UNOPTIMISED INTERMEDIATE CODE” string was commented out, so it was uncommented along with *dumpTAC(tac).* Also *tac = optimiseTAC(tac)* is commented out as therefore, this saves the unoptimized code to a .c file.

3. Below is a screenshot of the unoptimized intermediate code.



**Subtask 4 Testing.**

1. Below is the screenshot for testing point one. As && has higher precedence than || then the output should be 1. If the output did not satisfy the specification, then the output would be 0. Below is the screenshot for testing the point one. As && has higher precedence than || then the output should be 1. If the output did not satisfy the specification, then the output would be 0.  
   
2. Below is the screenshot for testing point two. Both the left and the right operand are false. Therefore, the output will be false.  
   
3. Below is the screen shot for testing point 3 and 4. As the first operand is true, all possible outcomes will be true. Therefore, babycino will not evaluate the right-hand side regardless of whether it is true or false. This also proves point 4.  
   