YOU MAY RECEIVE A ZERO IF ANY PART OF YOUR CODE IS TO SIMILAR TO ANOTHER STUDENTS PAST OR PRESENT

1. (3 points) Develop a class/struct for Token, **must contain**:
    a. String for lexeme representation
    b. Int for token code
2. (2 Points) Develop a **Compiler Class** that
    a. Has a method that takes in an ***INPUT FILE*** and converts it to one input str
3. (15 points) Develop a **Lexer** class: (Code 10, defintions 5)
    a. An instant of this class should exist In the **Complier Class**
    b. Takes in a string in its constructor
    c. Converts a string into a list of Token object if there exist no errors
        i. Should ignore block comments
        ii. Should ignore single line comments
    d. Should contain the following tokens and ***clear patterns or automata to recognize them*** (in the comments each should be specified)
        i. real_literal represents fractional number
        ii. natural_literal represent whole numbers and 0
        iii. bool_literal
        iv. char_liter represents a single ascii charater including escape character
            1. Java rules for escape character
        v. string_literal represents a any number of ascii charater including escape character
            1. Java rules for escape character
        vi. Keywords For
            1. Selection statement
            2. Loop Statement
            3. Variable declaration for
                a. Strings
                b. Naturals
                c. Character
                d. Reals
                e. Booleans
        vii. Special Symbols for
            1. Addition
            2. Subtraction
            3. Multiplication
            4. Division
            5. Exponentiation
            6. Symbol(s) to specify the breaking order of operations
            7. Greater than
            8. Less Than
            9. Greater Than or equal too (must be more than two characters)
            10. Less Than or equal too (must be more than two characters)

11. Equal to
12. Not equal too (must be more than two characters)
13. unary negation operator
14. Logical Not
15. Logical And
16. Logical Or
17. Symbol(s) of grouping code blocks
18. Parameter separator
19. Symbol(s) to specify the parameters of a function

viii. Variable/function identifier

4. (20 points) Develop a **Parser** class: Code 10 definitions 10
    a. An instant of this class should exist In the **Complier Class**
    b. Takes in aa list of Token object in its constructor
    c. Outputs a parse tree of called functions that would recognize the input is syntactically correct
    d. **REQUIRE YOU CREATE GRAMMAR RULES THAT WOULD SATISFY A TOP DOWN PARSER**
    e. Should be coded in the style of a recursive decent parser
    f. Code should be able to handle multiple code statements
        i. A valid code file should be able to have 0 or many valid statements
    g. A statement should be able to be one of the following
        i. Code block
        ii. Selection Statement
        iii. Loop Statement
        iv. Assignment Statement
            1. Should be able to assign an expression to a variable
            2. Expressions should be able to have Boolean solutions
            3. Operands in expressions can be variables, real_literal, natural_literal, bool_literal, charl_literal, string_literal, function_call
            4. Expressions should allow for unary negation operator
                a. If this symbol comes after any sumbol outside of the assignment operator or the opening symbol(s) to specify the breaking order of operations it should require it to be in the symbol(s) to specify the breaking order of operations
        v. Declaration statement
            1. Variables
            2. Functions Definition
5. (5 points) use Denotational semantics to define your selection statement
6. (5 points) use Denotational semantics to define your loop statement
7. (10 points) use Denotational semantics to define your Expr statement

8. (5 points) use Denotational semantics to redefine your Expr statement so it can return a Boolean solution
9. (10 points) Define the attribute grammar for your assignment statement, make sure it follows the following rules
    a. String + String does concatenation
    b. String * Natural repeats the Natural
    c. Assign bool to natural is allowed
    d. Assign natural to bool is allowed
    e. Assign char to natural is allowed
    f. Assign natural to char is allowed
    g. Assign natural to real is allowed
    h. No other types are allowed to be assigned to others outside of their own
    i. Dividing by zero is an error
    j. Modulo operating by zero is an error
10. (15 point) choose 3 syntactically valid assignment statements with at least 7 tokens to show these rules failing or passing semantic ruls
11. (10 points) Axiomatic Semantics (find the weakest precondition) :
    a.

```
a = 2 * (b - 1) - 1 {a > 0}
```

b.

```
if (x < y)
x = x + 1
else
x = 3 * x
{x < 0}
```

c.

```
y = a * 2 * (b - 1) - 1
if (x < y)
x = y + 1
else
x = 3 * x
{x < 0}
```

d.

```
a = 3 * (2 * b + a);
b = 2 * a - 1
{b > 5}
```