Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Escuela de Diseño, Ingeniería y Arquitectura
Departamento de Tecnologías de Información y Computación

Compilers Final Mock Exam

Instructor: Ariel Ortiz

INSTRUCTIONS

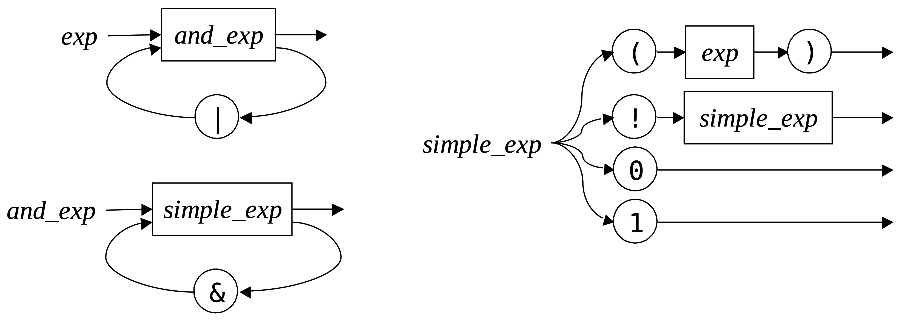
You must solve the following problem on your own using your mobile computer. You are allowed to use any information and/or software contained in your computer, as well as any written material (manuals, program listings, books, notes). During the exam you may not be connected to the network, speak to anyone, or use material belonging to somebody else.

The complete solution to the problem must be stored in one, and only one, source file called *`A0MMMMMMM.cs`* (where *`A0MMMMMMM`* is your student ID). Copy this file to the USB removable disk provided by your instructor and hand it in. Make sure the source file includes at the top the authors' personal information (name and student id) within comments.

Time limit: 180 minutes.

PROBLEM DESCRIPTION

Boolang is a small boolean expression language. The following is a syntax diagram for this language (where *exp* is the start symbol):



The language has the following semantics:

1. The symbols `0` and `1` represent the literal values of false and true, respectively.
2. The symbol `!` represents the conventional *not* prefix unary operator.
3. The symbol `&` represents the conventional *and* infix binary operator.
4. The symbol `|` represents the conventional *or* infix binary operator.
5. The precedence and associativity of the operators is defined in the syntax diagram itself.
6. Parenthesis may be used for grouping sub-expressions and forcing a specific evaluation order.
7. Any space and tab characters contained within an expression should be ignored.

Write a C# program that compiles a *Boolang* expression. For any given input, your program must parse it, build an abstract syntax tree, and generate CIL assembly language. Have this in mind:

- The input expression is always received as a command line argument.
- If a syntax error is detected, a "parse error" message should be displayed, and the program should end.
- If no syntax errors are detected, the AST should be displayed.
- The assembly language output must always be stored in a file called *`output.il`*.
- The purpose of the generated code is to evaluate at runtime the input expression and print the result to the standard output (issuing a call to the *`System.Console.WriteLine`* method from the CLI class library).
- The *`ilasm`* command will be called manually from the OS command line.

Example: When given the next command at the terminal:

```
Boolang.exe '!(0 | 0) & 1'
```

the following AST should be displayed at the standard output:

```
Program
  And
    Not
      Or
        Literal_0
        Literal_0
    Literal_1
```

and the contents of the produced *`output.il`* file should be:

```
.assembly 'boolang' {}

.class public 'FinalExam' extends ['mscorlib']'System'.'Object' {
    .method public static void 'start'() {
        .entrypoint
        ldc.i4.0
        ldc.i4.0
        or
        ldc.i4.1
        xor
        ldc.i4.1
        and
        call void ['mscorlib']'System'.'Console'::'WriteLine'(int32)
        ret
    }
}
```

Once assembled using *`ilasm`*, the above program outputs the following on execution:

```
1
```

Note that the assembly and method directives are always the same. Use this previous example to deduce by yourself the mapping between CIL instructions and the different *Boolang* language elements.

GRADING

The grade depends on the following:

1. **10** The C# code compiles without errors.
2. **30** Fulfills point 1, plus: performs lexical and syntax analysis without any errors.
3. **50** Fulfills points 1 and 2, plus: builds and prints the AST without any errors.
4. **100** Fulfills points 1, 2, and 3, plus: generates CIL assembly language without any errors.

"There are only two kinds of programming languages:
those people always bitch about and those nobody uses."

– Bjarne Stroustrup, creator of C++.