COMP-520 Midterm 2015

- This is a closed book exam, but you are provided with the summary sheets for the virtual machines and the flex/bison and sableCC3 examples for the *tiny* language.

  You will have one hour and 20 minutes to complete the exam. There are 50 points in total that are distributed as follows: Q1: 15 points, Q2: 15 points, Q3: 5 points, and Q4: 15 points. Make sure to leave adequate time to attempt all questions.

- Please write neatly, clearly, and concisely.

- On your answer book give your name, student ID and git group number.

1. (15 points) Assume that you have an alphabet of 'o', 'n', 'c', '<' and '>' (only these five letters). For each of the following languages specify one of the following:

   - If it is a regular language, give a regular expression that denotes the language.
   - If it is **not** a regular language, but it is a context-free language, give an **unambiguous** context-free grammar for the language.
   - If it is neither a regular, nor a context-free grammar, give a **context-free grammar** for a superset of the language, and then briefly describe a weeding routine that could be used to weed out all sentences that do not belong in the language.

   (a) (5 points) All sentences that do not contain the the '<' or '>' character.

   (b) (5 points) All sentences of a list of at least one tuple, where each tuple starts with '<' and ends with '>', and has only 'o', 'n' or 'c' in between. The empty tuple is also allowed ('<>').

   For example these sentences should be accepted: `<>`, `<con>`, `<><onc>`, `<nn><ooc><nnn>`.

   These should not be accepted: ϵ, `<<a>>`, `>a<`, `<<onc>>`.

   (c) (5 points) We wish to recognize all lists of at least one tuple, where the tuples are like in the previous part of this question, but with the additional requirement that each tuple is longer than the previous one.

   For example, `<><oo><nnn><onccc>` is a valid sentence because the tuples keep getting longer from left to right. The sentence `<oo><o>` is not valid because the 2nd tuple is shorter than the first. The sentence `<><oo><nnn>>` is invalid because it is not correct tuple syntax (extra '>' at the end).

`TURN THE PAGE OVER FOR MORE QUESTIONS ...`

2. (15 points) Based on the grammar for *tiny* as specified in the attached files, please answer the following questions.

   (a) (1 point) Are you basing your answer on the flex/bison code or the SableCC3 code?

   (b) (4 points) Draw the resulting AST that would be generated for the program:
   `(a + b) * c * d`

   (c) (4 points) Give the rightmost derivation for the same program, showing each step of the derivation, starting with the start production (`program` for the flex/bison example and `cst_exp` for the SableCC3 example):
   `(a + b) * c * d.`

   (d) (6 points) Let us define an extension of the tiny language called *tinyplus* which introduces an assignment statement, and a statement sequence (a sequence of one or more assignment statements). Each assignment statement must be terminated with a semicolon.
   A valid *tinyplus* program would look like:

   ```
   a = 3;
   b = 4 * a;
   c = a * b;
   ```

   A *tinyplus* program must be a statement sequence, and there must be at least one statement in the sequence. The right-hand side of an assignment statement is simply the current definition of expression in the given grammar.
   Describe the additions/changes that you must make to the flex/bison or sablecc3 scanner/parser specification to support this new language extension. You do not have to provide any extra code for building the AST, just provide the changes for the scanner rules and parser rules.

3. (5 points) Show the bytecode code, in Jasmin format, for the following small JOOS program. For each line of bytecode instructions, show the state of the expression (baby) stack. Be sure to indicate both the locals limit and the stack limit in the Jasmin code.

```
public static void m (int a, int b)
{  int c;
   c = a * a + b * b;
}
```

4. (15 points) At this point your group should have a very solid scanner, parser and type checker for your GoLite or OncoTime compilers. Answer the following, based on your group's implementation.

   (a) (1 point) Give your project type (OncoTime or GoLite), your implemetation platfors (C/flex/bison, Java/SableCC3, or whatever you are using) and your git group number.

   (b) (4 points) What has been the most challenging technical issue of your compiler to date? Explain the challenge and briefly discuss how you dealt with it.

   (c) (5 points) Show an example of a small GoLite or OncoTime program which demonstrates at least one nested scope (i.e. at least an outer and inner scope). Draw the top-most symbol table that you would have built upon exiting each scope. Make sure you clearly indicate where each exit scope occurs in your example program.

   (d) (5 points) If you were to add one language extension to your GoLite or OncoTime compiler, what would it be? For the GoLite students, you may give an extension that is not in Go. Explain the language feature, justify why it would be a useful addition, and **briefly** describe the changes that you would have to make to your compiler to add this feature.