CS712/CS812 Fall 2017 Final Exam

- **Place your answers on separate paper.**
- **The exam is closed book and notes.**
- **Please keep all electronic devices turned off and out of reach.**
- **This exam has 9 questions and a total of 100 points.**
- **For partial credit, show your work!**
- **When turning in your exam, please fold your papers in half, left-to-right, and write your name on the outside.**
- ***I will return your exam to your Kingsbury mailbox unless you tell me not to.***

Question 1 (10 points)

Compute the set of null-deriving nonterminals, the First sets and the Follow sets for the following grammar, in which S is the start symbol:

$S \to X\ Y$
$X \to a\ X$
$X \to$
$Y \to c\ X\ Z$
$Y \to b$
$Z \to a\ Z$
$Z \to a$

Question 2 (10 points)

Compute the LALR(1) configuration sets for the grammar of Question 1.

Question 3 (10 points)

Is the grammar of Question 1 LR(0)? Is it SLR(1)? Is it LALR(1)? Is it LR(1)? **Carefully state and justify each of your answers**.

Question 4 (10 points)

Compute the LL(1) Predict set for each production of the grammar of Question 1.

Question 5 (10 points)

Is the grammar of Question 1 LL(1)? Explicitly explain your answer in terms of the Predict sets.

Question 6 (**20 points**)
Consider the following Go program:

```
package main;

type S1 struct { x, y int }

var x *S1

var a int;

func main(){
  x = new(S1);
  a = x.x + 18 * 12;  // <-- trace compilation of this statement
  println(x.x);
}
```

Trace your compiler's processing of the second assignment statement in the function. By *trace* I mean: show the AST that would be produced by the parser, the AST after semantic analysis, and the code that would be generated. Be sure to explicitly show DEREF nodes and to label the types of expression nodes. You may discuss the generated code in terms of pseudo-code for LLVM.

Question 7 (10 points)
Here is the code for the main steps performed by our Go compiler to do semantic analysis and code generation:

```
SourceFile root = parser.getSemanticValue();
SymbolTable symbolTable = new SymbolTable();
UniverseBlock.initializePredeclaredTypes();
ProcessGlobalDefinitions processor =
  new ProcessGlobalDefinitions(symbolTable);
root.apply(processor);
UniverseBlock.insertPredeclaredIdentifiers(symbolTable);
Analyze analyzer = new Analyze(symbolTable);
symbolTable.completeGlobalDeclarations(analyzer);
root.apply(analyzer);
PrintWriter gener = openOutputFile(baseFileName, ".ll");
Encode genCode = new Encode(gener);
root.apply(genCode);
```

Please answer these questions about those steps:

- Explain what is stored in the symbol table.
- Explain what the *ProcessGlobalDefinitions* visitor does.
- Explain what the *completeGlobalDeclarations* method does.
- Why is the *insertPredeclaredIdentifiers* method called after the *ProcessGlobalDefinitions* visitor executes?

Question 8 (10 points)

Consider the following Java program fragment:

```
interface I {
  int meth1();
  int meth2();
}

interface I2 {
  int meth2();
  int meth3();
}

class A {
  public int i;
  public int j;
  public int meth1() { return 42; }
  public int meth2() { return 17; }
}

class B extends A implements I, I2 {
  public int j;
  public int meth2() { return 2; }
  public int meth3() { return 1066; }
}
```

Describe the layout of the VMTs generated to support the classes A and B. (Be sure to note that class B implements interfaces I and I2.) Use munged names to designate the function pointers being placed into a VMT. You can ignore methods inherited from the `Object` class.

Question 9 (10 points)

Consider the following Java code in the context of the class and interface definitions given in Question 8.

```
A a = new B();
B b = (B) a;
A a2 = new A();
I2 i2 = (I2) a2;
```

Show the layout of the object referenced by variable `a` prior to the first cast. What is the runtime algorithm used to detect that a cast to a class is illegal? Will the first cast cause a run-time error?

Show the layout of the object referenced by variable `a2` prior to the second cast. What is the runtime algorithm used to detect that a cast to an interface is illegal? Will the second cast cause a run-time error?