CS406: Compilers

Maximum Points: 100 Mid-semester examination 25/02/2020, 2:00PM to 4:00PM

Instructions:
The exam is open book, open notes (printed/written). No electronic devices allowed.
State your assumptions (if any) clearly.

Part I (short answers): *(20 points)*

1. Arrange the following phases of a compiler in order and mention which are part of the front-end: *Parser, Code generator, Scanner, Semantic Routines.* *(4 points)*

2. Explain briefly (in not more than a sentence) the difference between *syntax* and *semantics* of a language. (-*1 for writing more than a sentence*) *(2 points)*

3. The regular language *equivalent* to (0|1)*0(0|1)*
   i) (0|1)*(1|0)*
   ii) (1|0)*(10|00|0)(1|0)*
   iii)(1|0)*0(1|0)*
   iv)(1|0)*(1|0)(1|0)*
   a) iii only b) iii and iv only c)ii and iii only d) ii, iii, and iv *(4 points)*

4. a). Can the language $({}^{i}\,g\,)^{i}$, i ≥ 0 be recognized by an FSA? Why or why not? b) Can the language $({}^{k}\,g\,)^{k}$ *for one particular k* be recognized by an FSA? Why or why not? *(4 points)*

5. If I have a compiler for Intel chips, and Intel adds a new instruction to the x86 ISA, what phases of the compiler do I have to change?
   a) The parser b) The machine code generator c) The machine code generator and optimization passes d) None of the above answers are correct *(3 points)*

6. You want to replace the `do{…}while(cond);` construct of your programming language with `repeat{…}until(cond);`. The semantics remain the same. Your existing compiler uses 3 address code as its intermediate representation. What phase(s) of the compiler do you have to change?
   a) The scanner b) The parser and the scanner c) The machine code generator
   d) entire front-end *(3 points)*

Part II (Finite Automata): *(20 points)*

1. Draw an NFA for the regular expression: `((10)+|(00)+)*` *(6 points)*

2. Construct/draw a DFA for the NFA drawn previously. Use the state transition table method discussed in class. Show the state transition table. *(8 points)*

3. Reduce the DFA produced in the previous step (either using the split-node approach discussed in class, or by consulting the transition table constructed during previous step) *(6 points)*

Part III (Context-Free Grammar, Scanners, Parsers) *(38 points)*

Let G be the grammar:

```
S -> AB$
A -> xAz
A -> yAz
A -> λ
B -> Bz
B -> λ
```

1. What are the terminals and non-terminals of the grammar? *(6 points)*

2. Draw the parse tree for the string `yyxzzz$` *(6 points)*

3. Write the first sets for each non-terminal in the grammar *(6 points)*

4. Write the follow sets for each non-terminal in the grammar *(6 points)*

5. Write the Predict sets for each production in the grammar *(6 points)*

6. Fill in the entries of the parse table: *(6 points)*

	x	Y	Z	$
S				
A				
B				

7. Is the grammar LL(1)? Why or why not? *(2 points)*

Part IV (AST, Code generation) *(22 points)*

1. Draw the AST for the assignment statement a := b + c / d + 1 *(8 points)*

2. Give one possible three address code that would be generated for the above tree. Use the following instructions: LD A, T loads from variable A into temporary T. OP T1, T2, T3 performs T3 = T1 OP T2. ST T, A stores temporary into variable A. OP are ADD, DIV *(14 points)*