CS406: Compilers

Maximum Points: 15 Mid-semester examination 26/02/2021, 10:00AM to 12:00PM

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

Part I (short answers / multiple-choice based): ***(4 points)***

1. Arrange the following phases of a compiler in order (from front-end to back-end) and mention which are part of the front-end: Target-code optimization, Lexical analysis, Syntactic analysis, IR-code optimization, Semantic analysis. *(1 point)*

2. The regular language *equivalent* to (0|1)*1(0|1)*
   i) (01|11)*(1|0)*
   ii) (1|0)*(10|11|1)(1|0)*
   iii)(1|0)*1*(1|0)*
   iv)(1|0)*(1|0)(1|0)*
   a) ii only b) iii and iv only c) ii and iii only d) ii, iii, and iv *(1 point)*

3. Your language has a looping construct like C's `do-while` construct: `do{`S_1`;…;`S_n`;}while(cond`$_1$`);` Statements $S_1 \ldots S_n$ are executed once before evaluating the condition `cond`$_1$. The statements are executed repeatedly till the condition `cond`$_1$ becomes `false`.
   Pascal has the `repeat-until` construct: `repeat{`R_1`;…;`R_n`;}until(cond`$_2$`);` Statements $R_1 \ldots R_n$ are executed once before evaluating the condition `cond`$_2$. The statements are executed repeatedly till the condition `cond`$_2$, becomes `true`.
   Now, you want to *remove* the `do-while` feature in your language and *introduce* a `repeat-while` construct: `repeat{`T_1`;…;`T_n`;}while(cond`$_3$`);` Statements $T_1 \ldots T_n$ are executed once before evaluating the condition `cond`$_3$. The statements are executed repeatedly till the condition `cond`$_3$ becomes `false`.
   What phase(s) of the compiler you *must* change to implement the repeat-while construct? (explanation in support of your choices are welcome).
   a) The scanner b) The parser and the scanner c) The machine-code generator
   d) Parser, Scanner, and Semantic Routines *(1 point)*

4. Consider the grammar G_1:

```
1. S -> Ab$
2. A -> (bA)
3. A -> (A)
4. A -> x
```

   Draw the parse tree for the string: `(x)b$` (*1 point)*

CS406: Compilers

Maximum Points: 15 | Mid-semester examination | 26/02/2021, 10:00AM to 12:00PM

Part II (Context-Free Grammar, Parsers) ***(8.5 points)***

Consider the Grammar G_2:

1. S -> Ab$
2. A -> (bAb)
3. A -> (Ab)
4. A -> λ

5. Write the first sets for each non-terminal in the grammar *(2 points)*

6. Write the follow sets for each non-terminal in the grammar *(0.5 points)*

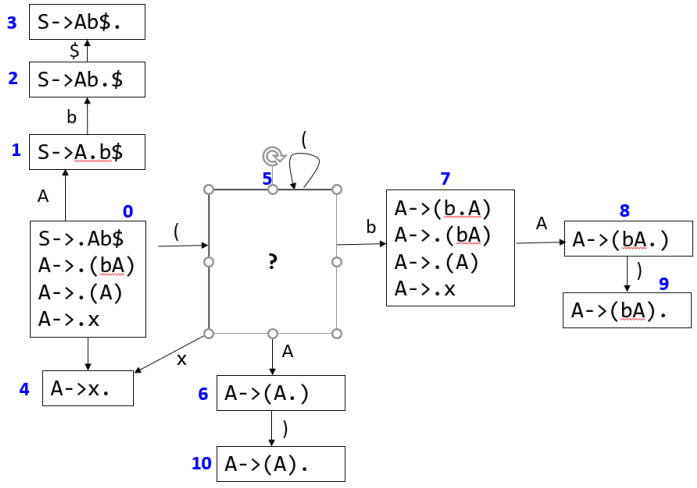
7. Write the Predict sets for each production in the grammar *(1.5 points)*

8. Fill-in the LL(1) parse table: *(1 point)*

	(	B	)	$
S				
A				

9. Is the grammar LL(1)? Why or why not? *(0.5 points)*

10. For the grammar G_1 in Part I, Q4, LR(0) CFSM is partially filled below. Fill in state 5 (draw only state 5): *(1.25 points)*



11. For the CFSM shown previously, parse stack contains: **0 5 4.** Next input token is **)**. Describe parser action(s). Use the format "Shift X" for shift actions, where X is the state being shifted to. "Reduce R, Goto X" for reduce actions where R is the production number and X is the state that parser winds up after reduction. Also provide the new state stack. *(1.75 points)*

Part III (AST) ***(2.5 points)***

12. Draw the AST for the following code snippet: `a := b + c * d + 1 ;` Traverse the AST in postorder and print the nodes. *(2.5 points)*