The American University in Cairo

Computer Science Department

CS 447 Final Exam – Spring 1999

Professor : Muhammed Mudawwar

Date : June 1st, 1999
Duration : 2 hours

1. (10 pts) Answer the following:

a) Discuss two approaches of entering record field names in a symbol table.
b) Discuss two symbol table implementations to handle scopes in a block-structured language.

2. Consider the following ambiguous grammar for expressions:

Expr → *Expr* **or** *Expr*
Expr → *Expr* **and** *Expr*
Expr → *Expr* **relop** *Expr*
Expr → *Expr* **addop** *Expr*
Expr → *Expr* **multop** *Exp*
Expr → **not** *Expr*
Expr → **(** *Expr* **)**
Expr → **intconst**
Expr → **realconst**
Expr → **boolconst**
Expr → **id**

Where **relop** represents one of the six relational operators "=", "<>", "<", "<=", ">", and ">=", **addop** represents "+" and "-", and **multop** represents "*" and "/".

a) (10 pts) Write a Yacc specification for the above grammar. Eliminate the ambiguity using the operator precedence and associativity rules assuming that the **or** operator has the least precedence and the **not** operator has the highest precedence.

b) (5 pts) In Yacc, is it possible to place operators with different associativity (left, right, and non-associative) at the same precedence level? Explain why such specification is allowed or disallowed?

c) (10 pts) Expressions are sometimes translated into Postfix notation. Add action rules to the above Yacc specification to translate expressions into postfix notation. For example, (*a* **or** *b* < *c* + *a* / *d*) * *c* translates into *a b c a d* / + < **or** *c* *. The postfix expression should be a string attribute to the non-terminal *Expr*.

3. Consider the following grammar:

(0) S' → S $
(1) S → **ID :=** A **;**
(2) A → **ID :=** A
(3) A → E
(4) E → E **+** P
(5) E → P
(6) P → **ID**
(7) P → **(** A **)**

a) (10 pts) construct the LR(0) Finite State Machine of the above grammar.
b) (7 pts) construct the LR(0) action and goto parsing tables. Is the grammar LR(0)?
c) (6 pts) construct the SLR(1) action and goto parsing tables. Is the grammar SLR(1)?

d) (7 pts) Using the SLR(1) table of part (c), trace the parse of **ID := ID := ID + (ID) ; $** by showing the content of the parse stack, remaining input and parser action at each step.

4. Consider the following LL(1) grammar G:

1: $E \rightarrow F\ R\ Q$
2: $Q \rightarrow \mathbf{+}\ F\ R\ Q$
3: $Q \rightarrow \lambda$
4: $R \rightarrow \mathbf{*}\ F\ R$
5: $R \rightarrow \lambda$
6: $F \rightarrow \mathbf{(}\ E\ \mathbf{)}$
7: $F \rightarrow \mathbf{id}$

a) (10 pts) Calculate the Predict sets for all productions and construct the LL(1) parsing table for grammar G. Use the production numbers specified above.
b) (8 pts) Write a recursive descent parser for grammar G. Four parsing routines are required. Assume the existence of a match routine, and a lookahead token.
c) (7 pts) Using a nonrecursive predictive parser, show the parsing of the input string **id*(id+id)$**. At each step show the content of the stack, remaining input, and parser action.

5. a) (4 pts) Show a DFA that corresponds with ((a | b)* (c | d)+) | aabb

b) (6pts) Show a DFA and write a regular expression for matching a Pascal-like comment delimited by (* and *). Individual *'s and)'s may appear in the comment body, but the pair *) may not.