Faculty of Arts & Sciences
Department of Mathematics & Computer Sciences
CMPS 274—Compiler Construction
Fall 2001–2002
Final Exam
Wednesday, February 6, 2002

Name:_____ **Student Id:**_____

Signature:_____

Instructions

- There are 5 problems and 20 pages. Make sure you have all of them.
- The exam is closed book, closed notes, and closed neighbor.
- Your handwriting should be readable so it can be graded. Include all work or justification for partial credit.

Problem 1	25 ✓		
1.a		2	
1.b		10	
1.c		4	
1.d		4	
1.e		5	
Problem 2	15 ✓		
2.a		3	
2.b		3	
2.c		3	
2.d		3	
2.e		3	
Problem 3	15 ✓		
3.a		10	
3.b		5	
Problem 4	31		
4.a		3 ✓	
4.b		3 ✓	
4.c		10	
4.d		15	
Problem 5	35		
5.a		5 ✓	
5.b		10	
5.c		5	
5.d		10	
5.e		5	
Total	121		

Problem 1 (2 + 10 + 4 + 4 + 5)

Consider the following grammar where lower-case bold letters are terminal symbols and upper-case letters are non-terminals:

1. S → **Aa**
2. S → **bAc**
3. S → **bc**
4. S → **bda**
5. A → **d**

(a) Compute the FIRST and FOLLOW sets of this grammar.

(b) Compute the LR(1) states clearly indicating the *kernel* and *non-kernel* items.

(c) Establish the SLR(1) parsing table.

(d) Establish the LALR(1) parsing table.

(e) Prove that the grammar is LALR(1), but not SLR(1) clearly indicating how the LALR(1) machine solves the problems associated with the SLR(1) machine.

Problem 2 (15% = 3 * 5)

Consider the following action-augmented grammar shown below. In this grammar, lower-case letters in bold are terminal symbols and upper-case letters are non-terminals..

1. S → TU { print("1"); }
2. S → UV { print("2"); }
3. T → **a**T { print("3"); }
4. T → **b** { print("4"); }
5. U → **c** { print("5"); }
6. U → **d**U { print("6"); }
7. V → **a** { print("7"); }

(a) Computer the FIRST sets for the above grammar.

(b) Compute the FOLLOW sets for the above grammar.

(c) Compute the SELECT sets for the above grammar.

(d) Construct the LL(1) parse table.

(e) Use the table to derive the output of the translation of the string "abcd".

Problem 3 (15%)

Given the following C-- program:

```
float myprog (int n)
{
  float x = 1.0;
  while (n > 0) {
    x = ((n > 5) || (n < 10)) ? 1.1 * x : 1.2 * x;
    n = n - 1;
  }
  return x;
}
```

(a) Draw the source tree for this program.

(b) Manually simulate semantic analysis and annotate the tree nodes with conversions and types.

Problem 4 (31%)

(a) What is the associativity of the assignment operator ('=') and the expression operators?

(b) What is the associativity of these operators in the rewritten grammar?

(c) If there are any differences, how do they manifest themselves in the source tree and how does your recursive parser deal with them?

(d) Assuming that the attribute of a symbol in the grammar is the corresponding node in the source tree, use the $-notation for attributes to augment the productions with actions that build the source tree.

Problem 5 (35%)

You have been hired for the summer as a programmer by the Tic-Tac-Toe Corporation. Among other things, this corporation processes student grades for the UAL university. Data files have the following format:

<student name> <student id> <course number> <semester taken> <grade received>.

Your job is to write a C program that reads the above data file and produces a grade report for each student in question. You are to store each student's grade report in a separate file that has the first four characters of the student's last name plus the last three digits of his/her id. For example, if a student's last name & id are Janko and 99123 respectively, then Janko's grade report should be stored in the file called "jank123.rpt". (You can ignore the case of characters.) Records in a file are to be sorted with respect to semesters. That is, Fall 92 comes before Spring 93. The description of the various fields are shown below together with sample input and output files.

Student name This field consists of a first name and a last name. The last and first names consist of a series of letters.

Student id This field is an integer field consisting of 6 digits. The first two digits designate the year in which the student entered the university; the last four are unique to the student—the university accepts at most 9999 students in a given year.

Course number This field is 6 character long string: three letters followed by three digits.

Semester This field consists of a *name* and a *year*. The name is one of four options: *Fall*, *Spring*, *Summer 1*, or *Summer 2*. The year is a two-digit field.

Grade This field consists of one character (A through F) possibly followed by a + or a –.

Sample input file

```
Zakaria Kicklikle 93123 CSC110 Fall 93 A.  Souraya A. Kicklikle
93124 CSC110 Spring 93 B.  Zakaria Kicklikle 93123 CSC112 Fall
94 D.  Fadi Nabil 93125 CSC110 Spring 94 C. Zakaria Kicklikle
93123 CSC210 Fall 94 C.  Souraya A. Kicklikle 93124 CSC300 Fall
94 B.  Fadi Nabil 93125 CSC499 Summer 1 94 C.  Zakaria
Kicklikle 93123 MTH300 Fall 92 A.  Souraya Kicklikle 93124
CSC210 Fall 93 B.
```

Sample output file

```
Grade Report
Name: Kicklikle, Zakaria
Id#: 93123

Course      Semester    Grade
------------------------------------------------------------
MTH300      F 92        A
CSC110      F 93        A
CSC112      F 94        D
CSC210      F 94        C
```

Questions

Having just taken the Compiler Construction course, you quickly realize that the above problem defines a *tiny* programming language and that your real task is to write a *compiler* for this tiny language. As such, you are planning to apply the compiler writing techniques you learned in the Compiler Construction course to help you organize your task thereby making your job easier :–). To develop your compiler, you will use the following plan in your development:

(a) Define the regular grammar used to recognize the tokens in the tiny programming language.

(b) Define a grammar for the above language.

(c) Classify the grammar you have defined and justify it.

(d) Assuming you have a parser generator tool that uses the *dollar* notation, augment the grammar with actions that will build the data structures needed to represent and collect student records.

Hint—design operations for initializing and building these data structures. Then, use these operations in the actions associated with the various productions. You do not need to implement these operations. However, you have to specify what these operations do.

(c) Augment the grammar to generate the desired output.

Hint—same as previous question.