University of Toronto

CSC 488/2107 Language Processors **Spring 1998**

Midterm [15% of final mark] **March 3, 1998**

Name (underline last name):

Student Number:

Instructions: This midterm is open book, open notes. It consists of 3 problems (8 pages). Write your answers clearly in the space provided. Total time is 50 minutes. Total marks - 100. Budget your time carefully. Good luck!

Problem 1 [30 marks total]. Regular expressions and finite-state automata.

1. Design a grammar for each of the following languages. Are these languages regular? Justify your answer.

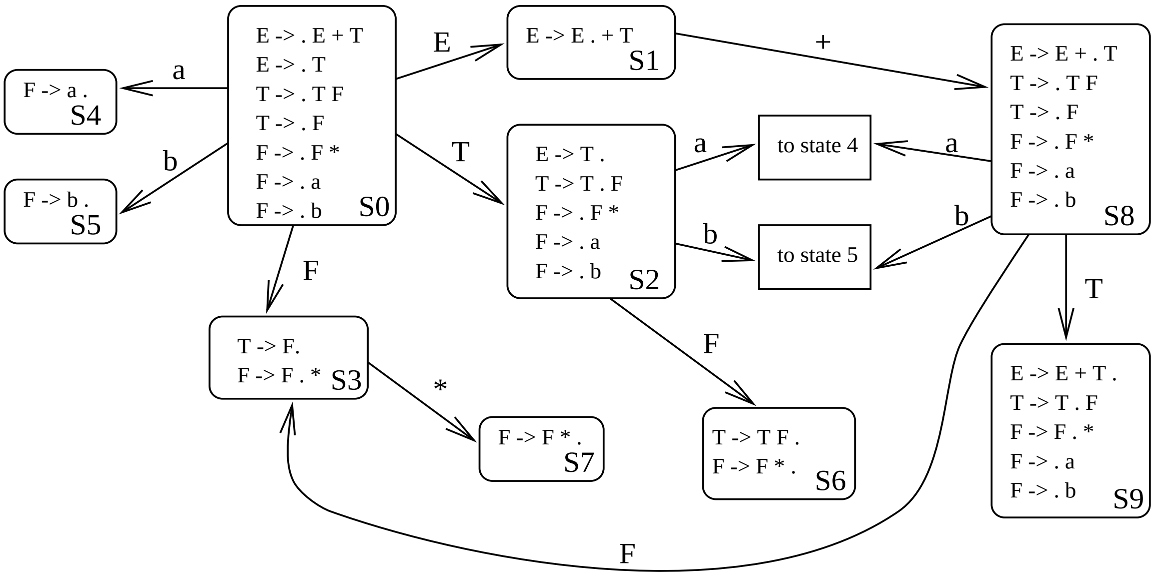
   (a) [10 marks] Strings of 0's and 1's in which 101 does not appear as a substring and which contain at most 3 0's.

   (b) [10 marks] Strings of 0's and 1's in the form $0^n1^{n+m}0^m$.

2. [10 marks] Let **Rev** be the operator that reverses strings. For example, **Rev**(abc) = cba. Let **R** be any regular expression. **Rev**(R) is the set of strings denoted by **R**, with each string reversed. Is **Rev**(**R**) a regular language? Justify your answer.

Problem 2 [40 marks total]. Grammars and parsing.

1. The following is a CFSM for a simple grammar G.



(a) [16 marks] Answer the following, justifying your answers:

- Is G LR(0)?
- Is G LALR(1)?
- Is G SLR(1)?
- Is G LR(1)?

A correct answer without an explanation is worth 1 mark for each question.

(b) [12 marks]. Show how a string

```
a + a b * *
```

is parsed. For every step of the parsing, show the remaining input and a stack of parser states. Also, indicate which action is used to move between the states.

2. [12 marks]. Show that there exist grammars that are SLR($k+1$) and LALR($k+1$) and LR($k+1$) but not SLR(k) or LALR(k) or LR(k), $k \geq 0$.

Problem 3 [25 marks]. Abstract syntax trees, semantic checking, symbol tables.
In the project, you are processing a very simple language, the one without any complex data structures like records, unions, pointers, etc. However, compilers for realistic languages need to take care of these language features. Design an abstract syntax tree capable of storing complex data structures and show the AST and the symbol table for the following C program:

```
#define ABOUNDH 5
#define ABOUNDL 7
double * findMem (float temp[]);
int abb[ABOUNDH][ABOUNDL], ** ipp, *(if());
const float Pi = 3.14;
typedef struct student {
  char* name;
  int abb;
  double if;
  union temp {
    int studentNumber;
    char* studentData;
  } unionRec;
} STUDENT;
STUDENT class[3], *studPtr;
```

Note: we should be able to understand the structure of your solution, without boring and tedious details.