University of Toronto

CSC 488/2107 Compilers & Interpreters **Spring 1999**

Midterm [15% of final mark] **March 9, 1999**

Instructions: This midterm is open book, open notes.
Non-programmable calculators allowed. No electronic communication devices allowed.
Total marks 100 - Total time is 50 minutes.
Two pages, 5 questions, **answer all questions**.

1. [20 marks]
In one programming language, bit string constants are defined according to the following grammar

bitStringConst	:	'"' bitGroupList '"' ;
bitGroupList	:	bitGroup
	|	bitGroupList bitGroup ;
bitGroup	:	binary , octal ;
binary	:	'(' '1' ')' binaryDigits ;
binaryDigits	:	binaryDigit ,
	|	binaryDigits binaryDigit ;
binaryDigit	:	'0'
	|	'1' ;
octal	:	'(' '3' ')' octalDigits ;
octalDigits	:	octalDigit
	|	octalDigits octalDigit ;
octalDigit	:	binaryDigit
	|	'2'
	|	'3'
	|	'4'
	|	'5'
	|	'6'
	|	'7' ;

Describe the design of a lexical analyzer for bit string constants that follow this definition. Describe carefully how the value of the bit string constant would be generated. You may assume that a bit string constant can contain at most 32 bits. Example bit string constants:

"(1)1010101(3)7651(1)11"
"(3)75707"
"(1)10101010101010101010101010101010101010101"

2 [20 marks]
In Fortran, arrays are laid out in memory in *column major order*, i.e. the array elements are placed in memory so that the *leftmost* subscript varies most rapidly. Devise an array subscripting algorithm for Fortran arrays (i.e. a mapping from an arbitrary array reference $A[E_1, E_2, ...E_n]$ to the address of the array element in memory).

3. [25 marks]
Describe the symbol and type table entries that a typical compiler would make for the following declarations:

```
#define LIMIT 1000000
const char names[] = { "compiler", "interpreter" } ;
float BigX[ LIMIT*LIMIT*LIMIT ] ;
typedef struct Node {
    char * names ;
    int marks[ 10 ] ;
    double rawMark ;
    struct {
        int teamNo ;
        float mark ;
    } assignments[ 6 ] ;
} StNode ;
StNode class[ 87 ] ;
StNode * classPrez, ** classList , *** facList ;
```

4. [15 marks]
List all of the semantic analysis checks that a typical compiler would perform when processing the declarations in Question 3.

5. [20 marks]
Given the LR(1) grammar:

Rule	Grammar
1	S $\rightarrow$ a A a
2	$\rightarrow$ b A b
3	$\rightarrow$ a B b
4	A $\rightarrow$ c
5	B $\rightarrow$ c b

Fill in the non-error entries (i.e. *shift*, *reduce by rule number* and *accept*) in the parse table for the grammar shown below.

Stack Configuration	Parse Table a	b	c	$\vdash$	State #
$\triangledown$					0
a $\triangledown$					1
b $\triangledown$					2
c $\triangledown$					3
A a $\triangledown$					4
B a $\triangledown$					5
A b $\triangledown$					6
b c $\triangledown$					7
a A a $\triangledown$					8
b A b $\triangledown$					9
b B a $\triangledown$					10
S $\triangledown$					11