0	00/
1	20/
2	15/
3	15/
4	20/
5	15/
6	15/
7	30/
8	20/
	150/

CMSC 331 First Midterm Exam

Name: _____

Student ID#:_____

You will have seventy-five (75) minutes to complete this closed book exam. Use the backs of these pages if you need more room for your answers. Describe any assumptions you make in solving a problem. We reserve the right to assign partial credit, and to deduct points for answers that are needlessly wordy.

0. Warm up (0 pts)

Four bugs sit at four corners of a one meter square. At the same instant all start walking at one meter per hour towards the bug on their clockwise direction and continually adjust their direction to move toward their target. How long does it take for them to meet and where? Extra credit: what will they do when they meet?

1. True/False (20 pts)

For each of the following questions, circle T (true) or F (false).

T F Lisp is most often used for Artificial Intelligence work.
T F FORTRAN was developed primarily for accounting and business applications.
T F The Ada programming language was designed as a standard language for DoD systems.
T F Java was one of the first object-oriented languages.
T F C# is a language developed by Microsoft and was based on Java and C++.
T F The BNF notation for syntax is more powerful than the EBNF notation.
T F Every attribute grammar can be rewritten as a BNF grammar with possibly more non-terminals and more rules.
T F Every BNF grammar G that has left recursion can be converted into a grammar G' that recognizes exactly the same language and has no recursive rules in it.
T F A grammar G is ambiguous if there is more than one parse tree for every sentence in the language defined by G.
T F If two languages have different non-terminal symbols, and different rules, the languages can not be T F In most programming languages that have infix operators, +, -, * and / are <u>left</u> associative.
T F A recursive descent parser for a language can not be based directly on a grammar that has recursion.
T F Axiomatic semantics associates pre-conditions and post-conditions with each statement in a program.
T F A variable's scope is the time during which the variable is bound to a storage location in memory.
T F An LR(k) parser always scans the input from left to right and finds k parses.
T F The idea behind denotational semantics is to define the meaning of statements in a programming language by translating them into statements in another language.
T F Static variables can not change their values.
T F A programming language is considered to be strongly typed if all type errors are detected at run time.
T F A lexical scanner reads a stream of tokens and produces a parse tree.
T F Every deterministic finite automaton (also known as a deterministic finite state machine) can be expressed as regular expression.

2. Short definitions (15 pts)

Give short (one to three sentences) answers to the following questions.

(a) What problems does a grammar with direct or indirect left recursion cause for a recursive descent parser?

(b) What is *left factoring* and why is useful?

(c) What does it mean for a set of grammar rules to be *pairwise disjointness*?

3. Regular expressions (15 points)

Briefly describe the language accepted by each of the following regular expressions: (5 each)

(a) 10* 1 | 01* 0

(b) 1 ((0 | 1)*)*

(c) (0 | 1 (0 | 1)*)

4. Simple Grammars (20)

Consider the following grammar where uppercase letters indicate non-terminals and lowercase letters indicate terminals:

```
S -> a S c B | A | b
A -> c A | c
B -> d | A
```

Which of the following sentences are in the language defined by this grammar? If the sentence is in the language, show a parse tree.

(a) aaccc

(b) ccc

(c) abcd

(d) accccc

5. Static and dynamic scope (15 pts)

K is a new programming language that is exactly like C, except it uses dynamic scoping for determining the binding of non-local variables rather than static (aka lexical) scoping. Consider the program to the right. What will it print when run when it's

(a) Compiled with a normal C compiler?

(b) Compiled with your new K compiler?

```
#include <stdio.h>

int i = 100;

void g(void) {
  printf("%d\n", i);
  i++;
  printf("%d\n", i);
}

int f(void) {
  int i=200;
  printf("%d\n", i);
  g();
  printf("%d\n", i);
  return(i);
}

int main(int argc,char **argv)
{
  printf("%d\n", f());
  printf("%d\n", i);
}
```

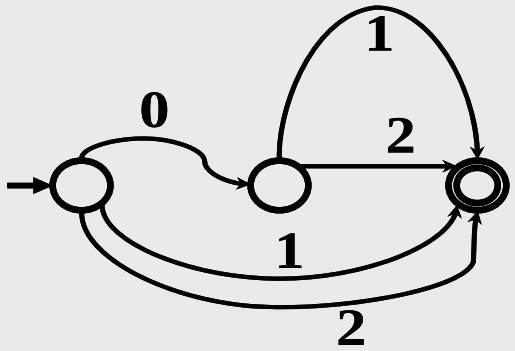
6. Axiomatic Semantics (15 points)

The following statement has a post-condition of `y<0`. What is the statement's weakest precondition?

```
if (x>y)
    then y := -x
    else y := x
```

7. Specifying a language (30 pts)

This problem asks you to define a simple language using a deterministic finite automaton (DFA), a regular expression (RE) and a context free grammar (CFG). Use the normal conventions for DFAs, REs and CFGs shown in the example to the right. The language is one for file names on Gizmodo, a new wrist watch computer under development. Here's an English description of the legal file names on a Gizmodo:



This example shows a DFA for the regular expression **`0?[12]`** and is also be described by the CFG:

```
S -> 0A | A
A -> 1 | 2
```

A Gizmodo file name starts with a prefix. It can optionally be followed by a dot and a suffix. A prefix must start with an alphabetic character which can be followed by any number of digits and/or alphabetic characters. The suffix, if there is one, must consist of exactly three alphabetic characters. Examples of legal filenames are `FOO`, `F32A`, `FOO.DOC`, `F.PDF` and examples of illegal filenames are `123`, `FOO.`, `BAR.EX`, `.XML` and `Q.123`.

To make it easier, assume that ALPHA represents any alphabetic character and DIGIT represents any digit.

(a) (10 pts) Draw a DFA for this language. Mark the accepting states with a double circle. Label each transition arrow with either an input character or one of the special tokens ALPHA or DIGIT.

(b) (10 pts) Write a RE to recognize this language using Lex's notation, again assuming that ALPHA represents any alphabetic character (e.g., [a-zA-Z] and DIGIT represents any digit (e.g., [0-9]).

(c) (10 pts) Recast your language as a CFG using the non-terminal FN as the start symbol and whatever other non-terminals you want. Again, treat DIGIT and ALPHA as tokens, so you need not expand them.

8. LR parsing (20 pts)

Consider an LR parser for the following simple expression grammar, producing the table to the right.

```
1: E -> E+T
2: E -> T
3: T -> T*F
4: T -> F
5: F -> (E)
6: F -> id
```

	Action						Goto		
State	id	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

(a) (1pt) What does an empty cell mean?

(b) (1pt) What symbol marks the end of the input?

(c) (1pt) Can this table be used to parse an input string of any length?

(d) (17 pts) Complete the following table which shows the first five steps of an LR parser parsing the input string **`(id+id)*id`**

STACK	INPUT	ACTION
0	(id+id)*id$	Shift 4
0 (4	id+id)*id$	