Final Exam Review Sheet
CS400 Spring 2013

General Instructions:

In addition to the material contained here, the following is part of this review by reference:

- Midterm Exam
- Midterm Exam Review Sheet
- All homework assignments

If you are comfortable with this material, you will do well on the final.

Because we only discussed selected items in the latter part of the course and skipped/skimmed over significant portions, for the material covered from Chapters 5, 6, 7, and 9 only those items touched on by this review sheet and/or HW#10 and/or HW#11 will appear on the final.

Remember that partial credit will be given when and where work is shown that makes the line of reasoning evident. Also remember that if no work is shown for a particular problem, then no partial credit will be awarded on that problem.

The answer list on the next page is THE answer list that will appear on the final. If you are familiar with the concepts associated with each answer, you should have no problems.

If you have questions about the meaning of any question, post a query on Piazza. I will not provide much information on how to solve a question, but I do want to make sure that the questions themselves are clear. I will be sending the exam to the printer well in advance of the Final Exam, so any comments that warrant tweaks to the exam must be received soon.

PART I – Multiple Choice

Choose the **best** option from the list below and enter the corresponding letter designation in the space provided **on this page**. Each answer may be used zero, one, or more times. Keep in mind that the question may or may not represent a defining relationship to the answer; in other words, the relationship may not be complete but, rather, more of an 'example' type relationship.

A. Closure
B. Lexeme
C. Syntactical analysis
D. First class function
E. Currying
F. Right-most derivation
G. Token
H. Sentence
I. Parser
J. Lexer
K. Finite automaton
L. Regular grammar
M. Context-free grammar
N. Pushdown automaton
O. Production rule
P. Higher order function
Q. Lexical scope
R. Regular expression
S. Pairwise-disjoint
T. Alphabet
U. Lexical analysis
V. Referential transparency
W. Semantics
X. Grammatical ambiguity
Y. Left-most derivation
Z. Dynamic scope

1)	6)	11)	16)	21)
2)	7)	12)	17)	22)
3)	8)	13)	18)	23)
4)	9)	14)	19)	24)
5)	10)	15)	20)	25)

PART 2

1) Why is a union used for yylval in some Bison/Flex files?

2) Why shouldn't the variable yytext be referenced in the Bison input file?

3)What is meant when Bison issues a shift/reduce, reduce/reduce, shift/shift conflict warning?

4) Describe the following attributes of a variable: Name, Address, Value, Type, Lifetime, and Scope.

5) Describe the advantages and disadvantages of static, implicit heap--dynamic, explicit heap-dynamic, and stack-dynamic (a.k.a., automatic variables).

6) What type of variables, in terms of lifetimes, are "needed" in order to support recursive functions? Explain why this is so.

7) Why are stack-dynamic variables generally referred to as "automatic" variables.

8) What is meant when a language is described as "strongly typed"?

9) What is meant by "dynamic type binding"? What are some of the advantages and disadvantages?

10) What is the difference between "load time" binding and "runtime" binding? Give an example of each.

11) What is meant by "type coercion"? Describe the difference between implicit and explicit coercion.

12) What is the difference between a formal parameter and an actual parameter?

13) What is the difference between a positional parameter and a keyword parameter? What are some advantages and disadvantages of each?

14) Strictly speaking, what is the difference between a procedure and a function?

15) What is meant by "pass-by-value"? What are the advantages and disadvantages?

16) In C, the statement `printf("%f", x);` works the same way regardless of whether x is a float or a double. Why?

17) In C, parameters are always passed by value. Yet the concept of "pass-by-reference" is generally used to describe how arrays and strings are passed to a function. Resolve the apparent discrepancy in terms of the difference between semantics and how semantics are implemented.

18) In C, when an array is passed to a function and the formal parameter is not declared merely as a pointer to the type of data stored in the array, the declaration must include the number of elements in each dimension except one, which may be left blank. Which dimension is optional and why is it not necessary?

19) What is printed by the C statement: **`printf("%s", &(1+2)["Hello World"]+3);`**

20) In C, how are arguments corresponding to an ellipsis as the formal parameter list handled?