CS 210 Fall 2020 Final Examination **Name: ____________________________________**

Answer all questions to the best of your ability. This is an open note, open book exam.

1. (20 points) Explain Flex's regular expression evaluation rules. How does yylex() evaluate its input against the list of regular expressions supplied by the Flex specification file? What does it do to decide which regular expression to use?

2. (30 points) Write a recursive ML function (norn L) that takes an arbitrary list L of strings and returns a tuple (referred to in this paragraph as result) of length two containing two sublists. For each element in L, if its string contents constitute a number it should be placed in the first sublist of result, while if it is not a number it should be placed in the second sublist of result. You are guaranteed (or may require) that L be a list. If L were empty, you would return a tuple of two empty lists. You may write and call helper functions. You may not declare any non-local variables.

   Example: (norn [“1”, “2”, “buckle my shoe”, “3”, “4”, “shut the door”]) would produce ([“1”, “2”, “3”, “4”], [“buckle my shoe”, “shut the door”])

3. (40 points) Write a Flex specification for all the literal constant values of the atomic/scalar types of Unicon, as described at unicon.org/book/ub.pdf Appendix A. Allow and discard whitespace as separators between literal values. Return a different integer code for each type of scalar value for which Unicon has literals. Do not allocate token struct pointers. Do not return or handle reserved words, keywords, punctuation, or operators. Do not let syntax or semantics elements in Appendix A distract you. Focus on regular expressions that describe, as correctly, completely, and readably as possible, the various kinds of words that appear in Unicon source code as literal (constant) values.

4. (10 points) Explain the difference between “methods” and “static methods” in Java. Under what circumstances do you write a “method” and under what circumstances do you write a “static method”?

5. (40 points) Write a Bison specification for a subset of ML consisting of function definitions and calls, lists, integers, variables and if-expressions. You should assume that you have a working yylex() function that you can call to provide you with the following tokens: FUN, '(', ')', '[', ',', ']', ':', COLONCOLON, '@', INTLITERAL, '+', '-', '*', '/', '=', VAL, IF, THEN, ELSE. You should add more terminal symbols to your grammar if you need to.

6. (30 points) Write an Unicon procedure replace(s1,s2,s3) that *generates* (Unicon-style) strings that consist of variants of s3 in which different instances of s1 are replaced by s2. For example,

   replace(“the”, “this”, “Watch the quick brown fox jump over the lazy yellow dog.”)

   would generate two results, namely

   “Watch this quick brown fox jump over the lazy yellow dog.” and
   “Watch the quick brown fox jump over this lazy yellow dog.”

7. (20 points) One of the big ideas in Java is exception handling. If you are writing some Java code that might result in a user-defined exception, you have two choices. Describe these two choices and give a Java code example of each.

8. (30 points) Write a Java class named Gosling (named after a famous programming language inventor James Gosling) that contains a member variable named covid that is a dynamically sized collection of strings, and a public method named give_virus(). Initialize the covid to contain the strings “cough” and “sneeze”, and write the body of give_virus() to return a random element from covid. Write a subclass of Gosling named BabyGoose with fields named nasty and nice. Initialize nasty and nice to an empty collection structure. Give class BabyGoose a public method know(name) that checks whether name is in the nasty or the nice collection; if nasty, know() returns “bad”, if nice, know() returns “good”. If it is not on the list, insert name into either the nasty (if the length of name is odd) or the nice (if the length of name is even) collection, and then return “bad” or “good” appropriately. Give class BabyGoose a public method give_virus(name) that returns the string “lump in your throat” if know(name) returns “bad”, and returns a random element from covid (“cough” or “sneeze”) if know(name) returns “good”.