Instructions
Answer question #1 and any three others. (If you answer more, only the first four will count.)
Point values are as indicated. Please try to make your answers neat and coherent. Remember, if we can’t read it, it’s wrong. Partial credit will be given, so try to put something down for each question (a blank answer always gets 0 points!).

1. (1 point)
What month is “April Fools Day” celebrated in?

2. (33 points)
Let Equal be the set of strings that contain an equal number of a’s and b’s (in any order). That is, Equal = {ab, ba, aabb, abba, abab, baab, baba, bbaa, ...}.

Is Equal a regular set? If it is, give a regular expression or finite automaton that defines it. If Equal isn’t regular, explain carefully why.

3. (33 points)
Let s be a string. Define insert#(s) to be a function that inserts a # into each possible position in s. If s is n characters long, insert# returns a set of n+1 strings (since there are n+1 places a # may be inserted in a string of length n).

For example, insert#(abc) = {#abc, a#bc, ab#c, abc#}, insert# applied to a set of strings is the union of insert# applied to members of the set. Hence insert#(ab, de) = {#ab, a#b, ab#, #de, d#e, de#}.

Let R be any regular set (my choice). Show that insert#(R) is a regular set.
4. Write regular expression definitions for the following token classes:

(a) (17 points)
Unsigned integer literals that represent integers evenly divisible by 4, with no leading zeroes. That is, 0, 8, 120, and 1000000 are allowed, but 1, 004, 007 and 123 are not allowed.

(b) (16 points)
A multi-line comment that begins with ///* and ends with */ */ and does not contain the triple */ */ anywhere within its text. Thus
///* Compute a = b/c*d /* from ref 1 */* */ is OK, but
///* ///* Nesting NOT allowed! */ */ */ is not allowed.

5. (33 points)
It is often the case that what looks like an undeclared identifier is actually a valid identifier that has been “misspelled.” If we could determine that an apparently undeclared identifier was “very close” to a single valid (properly declared) identifier, we might “repair” the misspelling to the valid identifier (with a suitable warning, of course).

Assume we try to repair only identifiers of three or more characters, and that an identifier $i_1$ is considered a possible misspelling of a valid identifier $i_2$ if $i_1$ can be converted into $i_2$ by adding, deleting or changing a single character. Thus given the valid identifier cat, we would recognize car (one character changed), scat (one character added) and at (one character deleted) as possible misspellings.

Assume we add a member function to our symbol table class

```java
String validId(String badId);
```

validId returns the text of a valid, already declared identifier if exactly one such identifier exists that differs from badId by exactly one character insertion, deletion or replacement. If more than one possible repair of badId exists, or no repair of badId exists, validId returns the null string.

Explain how validId might be efficiently implemented. (An algorithm that looks at all identifiers and tries to generate all possible repairs is certainly not efficient. A string turkey could never be repaired to duck, and should never even be considered.)

6. In C, C++ and Java, multi-line comments don’t nest. Thus the following character sequence is scanned as one comment, followed by the operators * and /:

```c
/* Let’s skip this:
   sum=sum+x; /* update running sum */
*/
```

(a) (7 points)
Explain why scanners have difficulty in recognizing and accepting nested comments.

(b) (6 points)
Are there any advantages to changing programming languages and their scanners to allow nested comments?

(c) (20 points)
How would you change your CSX scanner to allow nested C-like comments as illustrated above?