CSE 401 Midterm Exam 2/8/17

Name ____ ID # ___________________

There are 6 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed books, closed notes, closed electronics. Please turn off all cell phones, personal electronics, alarm watches, and pagers, and return your tray tables and seat backs to their full upright, locked positions. Sound recording and the taking of photographs is prohibited.

If you have a question during the exam, please raise your hand and someone will come to help you.

Please wait to turn the page until everyone is told to begin.

Score _________________

1 _______ / 14

2 _______ / 14

3 _______ / 12

4 _______ / 34

5 _______ / 16

6 _______ / 10

Question 1. (14 points) Regular expressions and DFAs. One format for writing dates gives the date first, a 3-letter month abbreviation in the middle, and the year last (for this problem we'll use 2-digit years to keep things shorter). Examples are `8feb17` (today), `31dec99`, `1jan00`, and `4jul76`. More specifically, a date is a string `ddmmmyy` or `dmmmyy`, where:

- `d` or `dd` is a date in the range 1-31 with no leading zeros,
- `mmm` is a month consisting of three lower-case letters
- `yy` is a year in the range 00 to 99.

To simplify the problem, any date in the range 1-31 is valid for any month (i.e., don't worry about trying to prevent dates like `30feb12`), and any 3-letter sequence of lower-case letters is valid for the month.

You must restrict yourself to the basic regular expression operations covered in class and on homework assignments: *r s*, *r* | *s*, *r**, *r*+, *r*?, character classes like [a-cxy] and [^aeiou], abbreviations *name*=*regexp*, and parenthesized regular expressions. No additional operations that might be found in the "regexp" packages in various Unix programs, scanner generators like JFlex, or language libraries.

(a) (6 points) Give a regular expression that generates all valid dates according to the above rules.

(b) (8 points) Draw a DFA that accepts all valid dates according to the above rules. (There is additional space on the next page for your DFA if it doesn't fit here.)

Question 1. (cont.) Extra space for your answer if needed.

Question 2. (14 points) Scanners and tokens. Just for fun, we ran our MiniJava scanner using a file containing the following C++ code fragment as input:

```
bool Thing::f(int x) {
  return this->val <= x;
}
```

Below, list in order the tokens that would be returned by a scanner for MiniJava as it reads this input. If there is a *lexical* error in the input, indicate where that error is encountered by writing a short explanation of the error in between the valid tokens that appear before and after the error(s) (something brief like "illegal character #" if a "#" was found in the file would be fine). The token list should include additional tokens found after any error(s) in the input. You may use any reasonable token names (e.g., LPAREN, ID(x), etc.) as long as your meaning is clear.

A copy of the MiniJava grammar is attached as the last page of the test. You may remove it for reference while you answer this question. You should assume the scanner implements MiniJava syntax as defined in that grammar, with no extensions to the language.

Question 3. (12 points) Ambiguity. Consider the following grammar:

P ::= V | $V\,X$
V ::= V `w` | ε
X ::= `w`

Is this grammar ambiguous? If so, give a proof that it is by showing two distinct parse trees, or two distinct leftmost (or rightmost) derivations, for some string. If not, give an informal, but precise argument why it is not ambiguous.

Question 4. (34 points) The you-can't-say-you-weren't-expecting-it parsing question. Here is a tiny grammar.

0. *S′* ::= *S* `$` (`$` represents end-of-file)
1. *S* ::= *A B*
2. *S* ::= `c`
3. *A* ::= `a`
4. *A* ::= ε
5. *B* ::= `b`
6. *B* ::= *B* `b`

(a) (12 points) Draw the LR(0) state machine for this grammar.

(b) (8 points) Compute *nullable* and the FIRST and FOLLOW sets for the nonterminals *S*, *A*, and *B* in the above grammar:

Symbol	nullable	FIRST	FOLLOW
S			
A			
B			

(continued on next page)

Question 4. (cont.) Grammar repeated from previous page for reference:

0. $S' ::= S$ `$`
1. $S ::= A\ B$
2. $S ::=$ `c`
3. $A ::=$ `a`
4. $A ::= \varepsilon$
5. $B ::=$ `b`
6. $B ::= B$ `b`

(c) (10 points) Write the LR(0) parse table for this grammar based on your LR(0) state machine in your answer to part (a).

(d) (2 points) Is this grammar LR(0)? Why or why not?

(e) (2 points) Is this grammar SLR? Why or why not?

Question 5. (16 points) LL parsing. Here is the grammar from the previous question, but without the extra *S′* ::= *S* `$` production that we added for the LR parser.

1. *S* ::= *A B*
2. *S* ::= `c`
3. *A* ::= `a`
4. *A* ::= ε
5. *B* ::= `b`
6. *B* ::= *B* `b`

Does this grammar satisfy the LL(1) condition? Give a technical justification for your answer. If it is not LL(1), change the grammar so that it is suitable for LL(1) parsing without changing the language that it generates.

Hint: It may save some time to compute the FIRST and FOLLOW sets requested in the previous question before working on this one.

Question 6. (10 points, 1 each) Different parts of the front-end of a compiler detect different errors in source programs. For each of the following possible errors, indicate which part of the front-end of a MiniJava compiler would detect the error by filling in the blank with `scan`, if the scanner would detect the error; `parse`, if the parser would detect the error; `sem`, if the semantics/type-checker phase would detect the error; or `can't`, if the front end of the compiler cannot or is not guaranteed to detect the error.

Hint: think carefully about what, exactly, each part of the front end of the compiler does. The MiniJava grammar is attached as the last page of this exam for reference. You may remove it from the exam if you wish.

________ Identifier `x` is not declared in the statement `x=17;`

________ Identifier `x` has type `boolean` in the statement `x=17;`

________ There is no <= operator in MiniJava

________ In the expression `i%j`, there is no `%` operator in MiniJava

________ Boolean values cannot be multiplied by integers (i.e., `4*true` is illegal)

________ Missing expression following `while` in `while ( ) x=x+1;`

________ # is not a legal character in a MiniJava program

________ `x` has the value -1 in the expression `new int[x]`

________ Missing object reference in a MiniJava method call (i.e., `f(17)` is not possible but `this.f(17)` is allowed).

________ In the method call `this.f(17)`, the class of the expression `this` does not contain or inherit a method `f` that has a single parameter of type `int`.