CSE 401/M501 18au Midterm Exam 11/2/18

Name _____ ID # _____

There are 7 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 write on the front of each page only. Those are the only pages that will be scanned for grading.

There is an extra blank page at the end of the exam you can use if your answer(s) do not fit in the space provided. Please indicate on the original page(s) if your answer(s) is(are) continued on that last page.

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

Score _____

1 _____ / 14

2 _____ / 8

3 _____ / 10

4 _____ / 8

5 _____ / 32

6 _____ / 14

7 _____ / 14

Question 1. (14 points) Regular expressions. We would like to process strings that represent file paths in a hypothetical file system. A file path consists of a drive specification, zero or more directories separated by backslashes, and a filename with a required file extension. The drive specification consists of a single uppercase letter followed by a colon and a backslash. Directory names and file names start with a single lowercase letter, and may be followed by any number of lowercase letters, numbers, or underscores. File extensions are exactly the same, but may not contain underscores. The file name is separated from the final directory name (if any) by a backslash, and the file name is separated from the file extension by a dot.

Some examples of valid file paths:

```
C:\folder\file.txt
A:\abc\temp_25\d.b4
Z:\midterm_solutions.docx
D:\marketing3.pptx
```

Some invalid file paths:

`B:\18au\scanner.pdf`	(names cannot start with numbers)
`M:\data.a_file`	(extensions cannot contain underscores)
`E:\\backups`	(directories names cannot be empty and file extensions are required)
`Z:\docs\README.txt`	(must use lower-case letters only)

As with homework problems, 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 are allowed.

Write your ….

answers on ….

the next page.

Remove this page from the exam and do not include it when you hand in your exam. It will not be scanned or graded.

Question 1. Write your answers here. Hint: it may be useful to work on the regular expression and the DFA parts simultaneously.

(a) (7 points) Give a regular expression (possibly with subexpressions if it makes things clearer) that generates all valid file paths according to the above rules.

(b) (7 points) Draw a DFA that accepts all valid file paths according to the above rules.

Question 2. (8 points) Tokens 'R Us. We've been having so much fun testing the MiniJava scanner that we decided to see what would happen if we used it to process the following Bash shell script:

```
# comment
while [ $# -gt 0 ]
do
  echo $*
  shift
done
```

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 @" where 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 or changes to the language.

Question 3. (10 points, 2 each) Ambiguity I. The following grammar is ambiguous:

A ::= *B* b *C*
B ::= b | ε
C ::= b | ε

To demonstrate this ambiguity we can use pairs of derivations. Here are five different pairs. For each pair of derivations, circle OK if the pair correctly proves that the grammar is ambiguous. Circle WRONG if the pair does *not* give a correct proof. You do not need to explain your answers.

(Note: Whitespace in the grammar rules and derivations is used only for clarity. It is not part of the grammar or of the language generated by it.)

(a) OK WRONG

```
A => B b C => b b C => b b b
A => B b C => B b b => b b b
```

(b) OK WRONG

```
A => B b C => b b C => b b
A => B b C => b C   => b b
```

(c) OK WRONG

```
A => B b C => b b C => b b
A => B b C => B b b => b b
```

(d) OK WRONG

```
A => B b C => b b C => b b
A => B b C => b b C => b b b
```

(e) OK WRONG

```
A => B b C => B b   => b b
A => B b C => B b b => b b
```

Question 4. (8 points) Ambiguity II. The following grammar is ambiguous. (As before, whitespace is used only for clarity; it is not part of the grammar or the language generated by it.)

P ::= `!` *Q* | *Q* `&&` *Q* | *Q*
Q ::= *P* | `id`

Give a grammar that generates exactly the same language as the one generated by this grammar but that is not ambiguous. You may resolve the ambiguities however you want – there is no requirement for any particular operator precedence or associativity in the resulting grammar.

Question 5. (32 points) The you're-probably-not-surprised-to-see-it LR parsing question. Here is a small grammar, complete with the extra *S*′::=*S*`$` rule to handle end-of-file in the generated parser.

1. *S*′ ::= *S* `$` (`$` represents end-of-file)
2. *S* ::= `a` *K*
3. *S* ::= *J S*
4. *S* ::= `b`
5. *J* ::= `a`
6. *K* ::= `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*, *J*, and *K* in the above grammar:

Symbol	nullable	FIRST	FOLLOW
S			
J			
K			

(continued on next page)

Question 5. (cont.) Grammar repeated from previous page for reference:

1. S' ::= S `$` (`$` represents end-of-file)
2. S ::= `a` K
3. S ::= J S
4. S ::= `b`
5. J ::= `a`
6. K ::= `b`

(c) (8 points) Write the LR(0) parse table for this grammar based on the LR(0) state machine in your answer to part (a).

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

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

Question 6. (14 points) LL grammars. Consider the following grammar:

P ::= *R* `b`
R ::= *P S* `a` | `c`
S ::= *P* | `b`

(a) (6 points) Demonstrate that this grammar does not satisfy the LL(1) condition. Hint: you may find it useful to compute nullable and the FIRST and FOLLOW sets for some or all productions, but you are not required to do so, as long as you can give a reason the grammar is not LL(1).

(b) (8 points) Rewrite the grammar so that it is LL(1). Your answer must generate the same language as the above grammar but it should not contain any conflicts that would violate the LL(1) condition.

Question 7. (14 points) Semantics. Suppose we have the following statement in a MiniJava (*not* full Java) program:

```
x = a+b.f(c);
```

(a) (6 points) Draw an abstract syntax tree (AST) for this statement in the blank space at the bottom of the page. You are not expected to remember the exact names of the classes or node types in the MiniJava AST package used in the project code, but your answer should have the appropriate level of abstraction and structural detail that is found there, and the nodes should have reasonable names. In case it is useful, remember that a copy of the MiniJava grammar is attached at the end of the exam.

(b) (8 points) Annotate your AST by writing next to the appropriate nodes the checks or tests that should be done in the static semantics/type-checking phase of the compiler to ensure that this statement does not contain any errors. You do not need to specify an attribute grammar – just show the necessary tests. If a particular test applies to multiple nodes you can write it once and indicate which nodes it applies to, as long as your meaning is clear and readable.

Extra space for answers, if needed. Please be sure to label which question(s) are answered here, and be sure to put a note on the question page so the grader will know to look here.