Exam for CSE 450 (2016)

Answer the questions in the spaces provided on the page. If you run out of room for an answer, continue on the back of the page.

- DO NOT OPEN THE EXAM UNTIL TOLD TO DO SO
- You only need to answer 5 of the 6 questions.
- On one of the questions make a large slash through it, which indicates that it should not be graded.
- On every page (including the first and last page), write your first and last name, before answering the question. Unnamed pages may be lost.
- If you start to answer a question and then change your mind, please cross out the attempt and write DO NOT GRADE across it.

https://xkcd.com/844/
Question 1: Regular Expression Matching .................................................. 4 points
For each of the following regular expressions, indicate which of the strings can be generated?

(a) (2 points) \(1*0+1?0\)
   - 0
   - 00
   - 1010
   - 00010
   - 10
   - 01110
   - 1100
   - 1

(b) (2 points) \((ab*)+c?\)
   - ac
   - ababab
   - aaabbb
   - c
   - abcd
   - abbaabbcc
   - bbc
   - ab+c

Points earned: __________ out of a possible 4 points
Question 2: Construct Regular Expression ........................................ 4 points

(a) (2 points) Design a regular expression that will identify all literal integers where the digits are in numerical order. Allow the empty string.
For example, 014457889 is allowed but 8953 is not.

(b) (2 points) Design a regular expression that will identify an integer value between 0 and 255. Your expression should identify all of these values with no extra leading zeros and no additional characters.

Good examples:
- 200
- 36
- 2
- 0
- 255

Bad examples:
- -1
- 3.14
- 450
- 007
- 256

Points earned: __________ out of a possible 4 points
Question 3: Construct Context Free Grammars ............................. 4 points
Design a context free grammar using the YACC syntax for the following problems.

Example YACC CFG rule: S : 'a' B 'c'

(a) (1 point) All possible sets of matched parentheses.

(b) (1 point) All strings consisting of a series of a's followed by a series of b's, where there are always more a's than b's.

(c) (2 points) All possible palindromes (strings that read the same forward or backward) over the alphabet {a, b, c}.

Points earned: ____________ out of a possible 4 points
Question 4: New Context Free Grammar.............................................. 4 points

You want to implement a new array literal in Tubular that generates an array based on a comma-separated series of one or more char literals or one or more val literals contained within "[" and "]". Examples of some legal arrays are:

[[ 1, 2, 3, 4, 5 ]]
[[1.4142, 4, 3.1416, 2.7183]] No spaces are required!
[[ 'G', 'o', 'o', 'd', '!' ]] Character arrays are okay!
[ [ 42 ] ] Minimum one entry; spaces okay!

Examples if ILLEGAL array are:

[[ ]] Must have at least one entry!
[[ 1, 2, '3', 4]] Cannot mix types!
[[ 1, 2, , 4]] Cannot have an extra comma!
[[ 1, 2, 4,]] Cannot have a trailing (or leading) comma!
[[ 1 + 2, 3]] Cannot have expressions!

Write a set of YACC-style syntactic rules that capture the grammar for array_literall. These are the only tokens you may use: CHAR_LITERAL, VAL_LITERAL,'[', ']' and ','.

Points earned: __________ out of a possible 4 points
Question 5: Tube Intermediate Code.................................................. 4 points
Given the source code, fill in the blanks in the Tube Intermediate Code generated by a compiler.

```plaintext
val x = 1;
val y = 99;
while (x < 5) {
    print(y);
    val y = random(x);
    print(y);
    if (y) break;
}
```

<table>
<thead>
<tr>
<th>command</th>
<th>arg1</th>
<th>arg2</th>
<th>arg3</th>
</tr>
</thead>
<tbody>
<tr>
<td>val_copy</td>
<td>1</td>
<td>s4</td>
<td></td>
</tr>
<tr>
<td>val_copy</td>
<td></td>
<td>s1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>s5</td>
<td></td>
</tr>
<tr>
<td>val_copy</td>
<td></td>
<td>s2</td>
<td></td>
</tr>
<tr>
<td>while_start_1:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val_copy</td>
<td>5</td>
<td>s6</td>
<td>s7</td>
</tr>
<tr>
<td>out_char</td>
<td>\n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td></td>
<td>s8</td>
<td></td>
</tr>
<tr>
<td>val_copy</td>
<td>s8</td>
<td>s3</td>
<td></td>
</tr>
<tr>
<td>out_char</td>
<td>\n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if_end_1:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>while_end_1:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Points earned: ____________ out of a possible 4 points
Question 6: Parse Tree ........................................... 4 points
Given the context free grammar:

\[
\begin{align*}
S & : A \ 'b' \\
A & : \ 'a' A \ 'a' \\
A & : \ 'b'
\end{align*}
\]

Draw the parse tree (using the CFG above) associated for the string: \textbf{aaabaaab}

Points earned: __________ out of a possible 4 points
Question 7: Abstract Syntax Tree .............................................................. 4 points
Below is an Abstract Syntax Tree of a Tubular expression statement.

(a) (2 points) What is a statement (in Tubular Source Code) that could have generated such a tree? (Assume that the identifiers are variables that have been previously declared as of type val.)

(b) (2 points) If the preceding statements were:

\[
\begin{align*}
\text{val } a &= 11; \\
\text{val } x &= 13; \\
\text{val } y &= -4; \\
\text{val } z &= 4;
\end{align*}
\]

What is the state of the 4 variables?

<table>
<thead>
<tr>
<th>a</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Points earned: __________ out of a possible 4 points
Question 8: Tube Code Assembly .......................................................... 4 points
For the Tube Intermediate Code below, convert each line to Tube Code Assembly.

(a) (1 point) \texttt{val	extunderscore{}copy} 5 s3

(b) (1 point) \texttt{add} s3 s2 s3

(c) (1 point) \texttt{ar	extunderscore{}get	extunderscore{}idx} a4 s3 s9

(d) (1 point) \texttt{ar	extunderscore{}set	extunderscore{}idx} a4 s2 s5

Points earned: __________ out of a possible 4 points
Question 9: Arrays and Memory......................................................... 4 points

Note: Both parts (a) and (b) read and write to the same memory at the bottom of the page.

(a) (1 point) For the memory representation below, write in the row ”New Value” any values that would change from executing the Tube Intermediate Code instruction. Memory location 0 stores the start of the free memory.

```
ar_copy a3 a4
```

(b) (2 points) For the same memory representation below, write in the row ”New Value” any values that would change from executing the series of Tube Code Assembly instructions.

```
val_copy 2 regB
load regB regC
val_copy '#' regD
store regD regC
jump_if_0 regC label
val_copy 4 regD
label:
load regC regA
store regA regB
```

(c) (1 point) Final value for registers after the above TCA instructions:

```
regA: __________
regB: __________
regC: __________
regD: __________
```

<table>
<thead>
<tr>
<th>Location</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Value</td>
<td>1007</td>
<td>'b'</td>
<td>1</td>
<td>1001</td>
<td>1004</td>
<td>0</td>
<td>\n’</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>New Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>1000</th>
<th>1001</th>
<th>1002</th>
<th>1003</th>
<th>1004</th>
<th>1005</th>
<th>1006</th>
<th>1007</th>
<th>1008</th>
<th>1009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Value</td>
<td>0</td>
<td>2</td>
<td>'z’</td>
<td>'y’</td>
<td>1</td>
<td>'a’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>New Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Points earned: __________ out of a possible 4 points
Question 10: Extra Credit: NFA

Create a Non-deterministic Finite State Machine for the following regular expression:

\[((ab)+(0*1))+\]

Points earned: __________ out of a possible 1 points
Question 11: Extra Credit: DFA ........................................... 1 points

(a) Convert the above DFA to a table. Use a slash to denote a halt on failure.

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Which strings does the DFA accept?

- 0
- 000
- 1010
- 00010
- 1000000
- 010101000100
- 000100
- 0001000

(c) What Regular Expression accepts the same strings as the DFA?

Points earned: ___________ out of a possible 1 points
First Name: ____________________________  
Last Name: ____________________________

Autumn 2016

If you have finished early, feel free to bring your exam to an instructor.  
Or you can draw a picture of your favorite Pokémon.  
Or you can write a haiku about your love of Abstract Syntax Trees.

<table>
<thead>
<tr>
<th>Normal Questions:</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Expression Matching</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Construct Regular Expression</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Construct Context Free Grammars</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>New Context Free Grammar</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Tube Intermediate Code</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Parse Tree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Abstract Syntax Tree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Tube Code Assembly</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Arrays and Memory</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>20</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extra Credit Questions:</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra Credit: NFA</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Extra Credit: DFA</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>2</strong></td>
<td></td>
</tr>
</tbody>
</table>

Points earned: ___________ out of a possible 0 points
Tube Intermediate Code Overview

Arguments
VAL: This argument uses a value; it can be a literal integer, a literal char, a label, or a scalar variable.
SCL: This argument must be a scalar variable (which will be written to)
ARR: This argument must be an array variable

Instructions
add [VAL:num1] [VAL:num2] [SCL:result]  sub [VAL:num1] [VAL:num2] [SCL:result]
mult [VAL:num1] [VAL:num2] [SCL:result]  div [VAL:num1] [VAL:num2] [SCL:result]

Apply the given math operation on num1 and num2, and place the answer in result.
val_copy [VAL:from] [SCL:to]  Copy the value from into the scalar variable to.
out_val [VAL:num]  Output the number represented by the argument.
out_char [VAL:char]  Output the character represented by the argument.

test_* [VAL:num1] [VAL:num2] [SCL:result]
Options are: test_less, test_gtr, test_equ, test_nequ, test_lte, or test_gte

Compare num1 and num2. Set result argument to 0 or 1 based on if condition is false or true.

jump [VAL:line]  Move the instruction pointer to line.
jump_if_0 [VAL:test] [VAL:line]  Move the instruction pointer to line if test is equal to zero.
jump_if_n0 [VAL:test] [VAL:line]  Move the instruction pointer to line if test is NOT equal to zero.

random [VAL:max] [SCL:result]  Generate random number 0 to (max - 1) and store in result.

ar_get_idx [ARR:array] [VAL:index] [SCL:result]
Look up index in array and store its value as result.
ar_set_idx [ARR:array] [VAL:index] [VAL:value]
Loop up index in array and set its value to value.
ar_get_size [ARR:array] [SCL:result]
Look up the size of array and store it in result.
ar_set_size [ARR:array] [VAL:new_size]
Resize array to new_size, copying over those elements in common.
ar_copy [ARR:array 1] [ARR:array 2]
Copy the contents and size of array 1 into array 2.

Labels
A label is a string of alphanumeric characters, beginning with a letter that is used to reference a line number elsewhere in the code. When a label is created, it must be placed at the beginning of a line of code and it must end with a colon (‘:’). A label will typically be used to indicate the end point in a jump command.

TubeCode Assembly Overview

TubeCode Assembly is very similar to the intermediate code, with a handful of changes
• The array_* instructions are not available.
• Scalar variables are not available, but eight registers (regA through regH) take their place.
• Three new instructions are available that allow you to interact with memory. They are:

load [VAL:from] [REG:to]  Load the value in memory position from into register to.
store [REG:from] [VAL:to]  Store the value in register from into memory position to.
mem_copy [VAL:from] [VAL:to]  Copy the value in memory position from into memory position to.