Exam for CSE 450 (2017)

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 7 questions.

• On two of the questions make a large slash through them, which indicates that they 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/303/
Question 1: Arrays and Memory.................................................40 points

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

(a) (25 points) For the memory representation below, write in the row "New Value" any values that would change from executing the Bad Lang instruction. Memory location 0 stores the start of the free memory.

```
ar_copy a3 a5
ar_set_size a5 s1
ar_set_idx a5 s4 7
```

(b) (10 points) For the same memory representation below, write in the row "New Value" any values that would change from executing the series of Ugly Lang instructions.

```
load 1 regD
load regD regC
mem_copy regC regD
val_copy 8 regA
sub regA regC regB
```

(c) (5 points) Final value for registers after the above Ugly Code instructions:

<table>
<thead>
<tr>
<th>regA:</th>
<th>regB:</th>
<th>regC:</th>
<th>regD:</th>
</tr>
</thead>
</table>

<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>116</td>
<td>2</td>
<td>114</td>
<td>113</td>
<td>1</td>
<td>107</td>
<td>'c'</td>
<td>-9.5</td>
<td>3</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>112</th>
<th>113</th>
<th>114</th>
<th>115</th>
<th>116</th>
<th>117</th>
<th>118</th>
<th>119</th>
<th>120</th>
<th>121</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Value</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</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 40 points
Question 2: Compile Good To Bad .................................................. 40 points
Given the source code, fill in the blanks in the Bad Code generated by a compiler.

```
string x = "'"';
x.resize(random(100));
val z = 0;
while (z < x.size()) {
    print(x[z]);
}
```

<table>
<thead>
<tr>
<th>command</th>
<th>arg1</th>
<th>arg2</th>
<th>arg3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar_set_size</td>
<td>a2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar_set_idx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ar_set_idx</td>
<td></td>
<td>a2</td>
<td>a0</td>
</tr>
<tr>
<td>val_copy</td>
<td>100</td>
<td>s3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>s3</td>
<td>s4</td>
</tr>
<tr>
<td>ar_set_size</td>
<td>a0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>val_copy</td>
<td>0</td>
<td>s5</td>
<td></td>
</tr>
<tr>
<td>val_copy</td>
<td>s5</td>
<td>s1</td>
<td></td>
</tr>
<tr>
<td>ar_get_size</td>
<td>a0</td>
<td></td>
<td>s6</td>
</tr>
<tr>
<td>test_less</td>
<td>s1</td>
<td></td>
<td>s7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s7</td>
<td></td>
</tr>
<tr>
<td>ar_get_idx</td>
<td>a0</td>
<td>s1</td>
<td>s8</td>
</tr>
<tr>
<td>out_char</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td></td>
<td>while_start_0</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 40 points
Question 3: Deterministic Finite Automata .................................................. 40 points

(a) (20 points) For the following Deterministic Finite Automata (DFA), write a Regular Expression that accepts the same language.

(b) (5 points) Convert the above DFA to a table. Use a slash to denote a halt on failure.

(c) (15 points) Which strings does the DFA accept?

- bb
- babbcc
- bbbbcc
- babbabbabbbc
- babbbabbabbb
- babbbabb
- babbbbcc
- babbcc
- babcc

Points earned: ___________ out of a possible 40 points
Autumn 2017

First Name: ____________________________

Last Name: ____________________________

Question 4: Pair Literal .......................................................... 40 points

You want to implement a new pair literal in Good Lang that generates a pair of values (called left and right) contained within less-than and greater-than operators and separated by a comma. The types of left and right can be val literals, char literals, or pair literals. However, the type of left and right must not match. Examples of some legal pair literals are:

<5, 'c'>  
Nice and simple pair literal!

<6, <'d', 0>>  
Pairs are allowed in pairs!

<'y', <<'o', <'8', 8>>, 9>>  
You can nest as deep as you like!

Examples of ILLEGAL pair literals are:

<>  
Pairs can’t be empty!

<7>  
Pairs must have a left and right value!

<'c', 'y'>  
Left and right can’t have the same type!

<<1, 'K'>, <'G', 7>>  
Can’t match types (both left and right are pairs)!

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

Points earned: ___________ out of a possible 40 points
Question 5: Context Free Grammar Matching ................................................... 40 points
For each of the following Context Free Grammars (CFGs), indicate which of the strings can be generated? Note: terminals are lowercase letters and non-terminals are uppercase letters.

(a) (20 points)
   \[ S : A \ B \]
   \[ A : b \mid c \mid t \]
   \[ B : C \ A \mid B \ B \]
   \[ C : a \mid e \mid i \mid o \mid u \]
   - ○ CAB
   - ○ cat
   - ○ toot
   - ○ cucuc
   - ○ tubui
   - ○ betot
   - ○ bewit
   - ○ cattab
   - ○ tettob
   - ○ tebicot

(b) (20 points)
   \[ S : A \ S \mid B \]
   \[ A : h \mid k \]
   \[ B : A \mid w \mid C \ C \]
   \[ C : p \mid d \]
   - ○ w
   - ○ k
   - ○ pd
   - ○ kwh
   - ○ kkdd
   - ○ kkhp
   - ○ hhkhw

Points earned: __________ out of a possible 40 points
Question 6: Construct Regular Expression ................................................. 40 points

(a) (20 points) Design a regular expression that will match all positive even integers (no leading zeros) and nothing more.
For example, 15048 and 9860 are matched but 04 and 123407 are not.

(b) (20 points) Design a regular expression that will identify strings with at least one consecutive repeated letter. For simplicity, the language is restricted to the letters a, b, and c.
Matching examples:
- aa
- abcabba
- aabbccacbabb
- abb

Non-matching examples:
- a
- ababa
- abcchabca

Points earned: _________ out of a possible 40 points
Question 7: Bad to Ugly .................................................. 40 points
For each Bad instruction below, convert to Ugly Code. For this page only, you can not use the mem_copy command.

(a) (10 points) val_copy s10 s40

(b) (10 points) random 5 s9

(c) (10 points) ar_get_size a8 s99

(d) (10 points) ar_set_idx a19 2 \n'

Points earned: __________ out of a possible 40 points
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>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrays and Memory</td>
<td>40</td>
</tr>
<tr>
<td>Compile Good To Bad</td>
<td>40</td>
</tr>
<tr>
<td>Deterministic Finite Automata</td>
<td>40</td>
</tr>
<tr>
<td>Pair Literal</td>
<td>40</td>
</tr>
<tr>
<td>Context Free Grammar Matching</td>
<td>40</td>
</tr>
<tr>
<td>Construct Regular Expression</td>
<td>40</td>
</tr>
<tr>
<td>Bad to Ugly</td>
<td>40</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>200</strong></td>
</tr>
</tbody>
</table>

Points earned: __________ out of a possible 0 points
Bad Lang 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: array1] [ARR: array2]  Copy the contents and size of array1 into array2.

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.

Ugly Lang Overview

Ugly Lang is very similar to Bad Lang (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.