First Name: ______________________

Last Name: ______________________

PID (on your ID card): ______________

MSU Net ID: ______________________

Actual Exam for CSE 450 (2018)

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 them, which indicates that they should not be graded.

• If you start to answer a question and then change your mind, please cross out the attempt and write \textit{DO NOT GRADE} across it.

• Have your ID ready when you turn in your exam.
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 LMAOcode instruction. Memory location 0 stores the start of the free memory.

<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>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Value</td>
<td>87</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>63</td>
<td>0</td>
<td>81</td>
<td>‘b’</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>New Value</td>
<td>93</td>
<td>1</td>
<td>‘@’</td>
<td>60</td>
<td></td>
<td>87</td>
<td>7</td>
<td>‘b’</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

(b) (15 points) For the same memory representation below, write in the row “New Value” any values that would change from executing the series of ROFLcode instructions.

LOAD 80 regB
STORE 60 3
MEM_COPY regB 8
STORE regB regB
VAL_COPY 8 regA
SUB 10 regA regA
STORE ‘@’ regA

<table>
<thead>
<tr>
<th>Location</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>63</th>
<th>64</th>
<th>65</th>
<th>66</th>
<th>67</th>
<th>68</th>
<th>69</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>‘*’</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>

<table>
<thead>
<tr>
<th>Location</th>
<th>80</th>
<th>81</th>
<th>82</th>
<th>83</th>
<th>84</th>
<th>85</th>
<th>86</th>
<th>87</th>
<th>88</th>
<th>89</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Value</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</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>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Key:
-3 per extra mem location edited grouped to part a of b as fitting a.

6 points (Correct Location 9)
AR_GET_SIZE a4 s9

10 points (Correct Location 6, 87, and 0 Points 4,4,2 respectively)
AR_SET_SIZE a6 s9

6 points (Correct location 1)
ADD s8 s3 s1

3 points (Correct Location of 89)
AR_SET_IDX a6 s1 s9

b.

LOAD 80 regB

4 points (Correct Location 3)
STORE 60 3

3 points (Correct Location 8)
MEM_COPY regB 8

3 points (Correct Location 7)
STORE regB regB

5 points (Correct Location 2)
VAL_COPY 8 regA
SUB 10 regA regA
STORE @ regA
Question 2: Multidimensional Arrays ................................................. 40 points

Many languages allow for multidimensional arrays (also called arrays-of-arrays), but out of simplicity, I’ve not had you implement them in the projects. But I didn’t say anything about on the exam.

You want to implement a new literal in LOLcode that represents 2 dimensional arrays of TROOF literals. These arrays will be used to implement complicated binary logic.

Examples of some legal 2-dimensional array literals are:

- \([\text{WIN}]\) 2D array can hold TROOF literals
- \([\text{WIN}, \text{FAIL}]\) 2D array can also hold arrays of TROOFs
- \([\text{WIN}, \text{FAIL, WIN}]\) 2D arrays don’t need to be regular
- \([\text{WIN}, \text{FAIL, WIN, WIN}, \text{WIN}]\) 2D array can have many elements

Examples of **ILLEGAL** 2-dimensional array literals are:

- \([\text{}]\) 2D arrays can’t be empty
- \([\text{WIN}, []]\) 2D arrays can’t have empty arrays as elements
- \([4]\) 2D arrays elements can’t be NUMBR literals
- \([\text{WIN} \text{FAIL}]\) Commas are not optional
- \([\text{WIN} \text{FAIL}, \text{FAIL,}]\) No trailing comma
- \([\text{WIN}, \text{WIN}, \text{FAIL}]]\) Can have at most 2 dimensions (not 3)

Write a set of Context-Free Grammar rules that capture the grammar for 2d_array_literal. The only terminal symbols you may use are: LETTR_LITERAL, NUMBR_LITERAL, TROOF_LITERAL, OPEN_BRACKET, CLOSE_BRACKET, COMMA and EPSILON.

Solution:

\[
\begin{align*}
2d\_array\_literal & \rightarrow \text{OPEN\_BRACKET} \ 2d\_body \ \text{CLOSE\_BRACKET} \\
2d\_body & \rightarrow \ 2d\_element \\
2d\_body & \rightarrow \ 2d\_element \ \text{COMMA} \ 2d\_body \\
2d\_element & \rightarrow \ \text{TROOF\_LITERAL} \\
2d\_element & \rightarrow \ 1d\_array \\
1d\_array & \rightarrow \ \text{OPEN\_BRACKET} \ 1d\_body \ \text{CLOSE\_BRACKET} \\
1d\_body & \rightarrow \ \text{TROOF\_LITERAL} \\
1d\_body & \rightarrow \ \text{TROOF\_LITERAL} \ \text{COMMA} \ 1d\_body
\end{align*}
\]
5: Start rule correct (I also give "S" and "Start" 5 points, "array" 4 points, and ridiculous stuff 0 points)

15: Internal arrays
10: Internal arrays allow 1 or more elements (not 0 or more)
5: Comma usage correct.
5: Troofs are the only base type allowed.
15: 2D arrays
5: [] prohibited
5: 3D array prohibited
5: Everything else good.
Question 3: Non-deterministic Finite Automata.................................................40 points

(a) (10 points) Fill in the bubbles of the strings that the above Non-deterministic Finite Automata (NFA) accepts?

\[ \checkmark \quad 1 \]
\[ \quad 0 \]
\[ \checkmark \quad 100 \]
\[ \quad 001 \]
\[ \quad 101001 \]
\[ \checkmark \quad 1111 \]
\[ \quad CB \]
\[ \quad \varepsilon \]

(b) (25 points) Convert the above NFA to a Deterministic Finite Automata (DFA).

Solution:

(c) (5 points) Convert the DFA you created above into to a table. Use a slash to denote a halt on failure. Note: There may be additional rows that you may not need for your solution.

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>/</td>
<td>CD</td>
</tr>
<tr>
<td>CD</td>
<td>BD</td>
<td>C</td>
</tr>
<tr>
<td>BD</td>
<td>BD</td>
<td>/</td>
</tr>
<tr>
<td>C</td>
<td>/</td>
<td>C</td>
</tr>
</tbody>
</table>
3.a: 6 points for all but one correct
3 points for all but 2 correct
0 points for anything else or if epsilon or nonterminals were selected

3.b:
4 segments
*5 points alloted for correct start and terminals 2 for a correct start and 3 points
*5 points for it being a dfa also its a zero on the question if this is not the case
*5 points if it includes the set of answers even if it is not restrictive enough
*10 points for correctness ingoring the first 5 points checks
*5 points if it includes the set of answers even if it is not restrictive enough

3.c:
Ok to account for variance in part b
5*correct transitions/ set of all correct transitions.
Question 4: Parsing Python Code .................................................. 40 points

x = 5
print("x = ", x)
while 7:
    print(10 - x)
    x = x + 1
    if x > 20:
        break

(a) (5 points) How many symbols are added to the symbol table when the Python source code above is compiled?

1

Key: 2 points for the answer 2 (mistaking assignment for variable declaration)

(b) (5 points) How many literals are present in the source code?

6

Key: 3 points for 5 or 1 (forgetting Integer or String Literals)

(c) (30 points) Draw an abstract syntax tree which may be generated by parsing the source code.
Key:  
A: 5 points for correct root node (with three children)  
B: 5 points for while node having two children  
C: 5 points for working assign-add  
D: 5 points for print having two and one children  
E: 10 points for otherwise correct
Question 5: AR_SET_SIZE_ZEROES ................................................. 40 points

AR_SET_SIZE is used to create an array of a particular size at the start of free memory. Normally, this command doesn’t adjust any memory associated with the space set aside for holding the elements of the array. The interpreter provided for LMAOcode and ROFLcode guarantees that all memory positions initially hold a zero, until changed to other values by the program. However, my boss just told me that needing to ensure that memory was zero’ed out before the program could use it is too slow, as most programs don’t rely on that feature. So, we removed that feature, meaning that memory initially is filled with garbage/random values. But, we were asked to add a modified version of AR_SET_SIZE (named “AR_SET_SIZE_ZEROES”) to LMAOcode that performs the same role of AR_SET_SIZE and then also ensures that all the elements of the array are initialized to zero.

Write the ROFLcode instructions that should be added to the AR_SET_SIZE instructions below that performs “AR_SET_SIZE_ZEROES a99 s66”. Be sure to add comments denoting what registers and instructions are performing which roles so we can award partial credit if needed.

LOAD 0 regA
STORE regA 99
LOAD 66 regB
STORE regB regA
ADD regA regB regC
ADD 1 regC regC
STORE regC 0
# What instructions should be added here?

Solution:

zero_start_1: # This code initializes the zeroes
ADD 1 regA regA
# Make pointer (regA) point at the next element
TEST_GTE regA regC regD
# Test if the pointer is past the end of the array
JUMP_IF_NO regD zero_end_2 # If so, jump to end of zero_loop
STORE 0 regA # Zero that element of the array
JUMP zero_start_1 # Loop to start next loop
zero_end_2: # All done
Question 6: Compile LOLcode To LMAOcode .................................................. 40 points
Given the source code, fill in the blanks in the LMAOcode generated by my compiler.

HAI 1.450
I HAS A dog ITZ LOTZ A TROOFS AN THAR IZ 2
O RLY? SAEM GIMMEH ’A’
YA RLY
   VISIBLE IN dog’Z 1 PUT FURSTBIGGR 10 WHATEVR
NO WAI
   I HAS A dog ITZ A NUMBR AN ITZ 4
   VISIBLE dog!
OIC
KTHXBYE

<table>
<thead>
<tr>
<th>command</th>
<th>arg1</th>
<th>arg2</th>
<th>arg3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAL_COPY</td>
<td>2</td>
<td>s2</td>
<td></td>
</tr>
<tr>
<td>AR_SET_SIZE</td>
<td>a1</td>
<td>s2</td>
<td></td>
</tr>
<tr>
<td>IN_CHAR</td>
<td>s3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAL_COPY</td>
<td>‘A’</td>
<td>s4</td>
<td></td>
</tr>
<tr>
<td>TEST_EQU</td>
<td>s3</td>
<td>s4</td>
<td>s5</td>
</tr>
<tr>
<td>JUMP_IF_0</td>
<td>s5</td>
<td>label_a</td>
<td></td>
</tr>
<tr>
<td>VAL_COPY</td>
<td>10</td>
<td>s6</td>
<td></td>
</tr>
<tr>
<td>RANDOM</td>
<td>s7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST_GTR</td>
<td>s6</td>
<td>s7</td>
<td>s8</td>
</tr>
<tr>
<td>VAL_COPY</td>
<td>1</td>
<td>s9</td>
<td></td>
</tr>
<tr>
<td>AR_GET_IDX</td>
<td>a1</td>
<td>s9</td>
<td>s10</td>
</tr>
<tr>
<td>AR_SET_IDX</td>
<td>a1</td>
<td>s9</td>
<td>s8</td>
</tr>
<tr>
<td>VAL_COPY</td>
<td>s8</td>
<td>s10</td>
<td></td>
</tr>
<tr>
<td>OUT_NUM</td>
<td>s10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUT_CHAR</td>
<td>‘\n’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JUMP</td>
<td>label_b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>label_a:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAL_COPY</td>
<td>4</td>
<td>s11</td>
<td></td>
</tr>
<tr>
<td>VAL_COPY</td>
<td>s11</td>
<td>s12</td>
<td></td>
</tr>
<tr>
<td>OUT_NUM</td>
<td>s12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>label_b:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>