CS314 - Principles of Programming Languages
Practice Final
Fall 2019

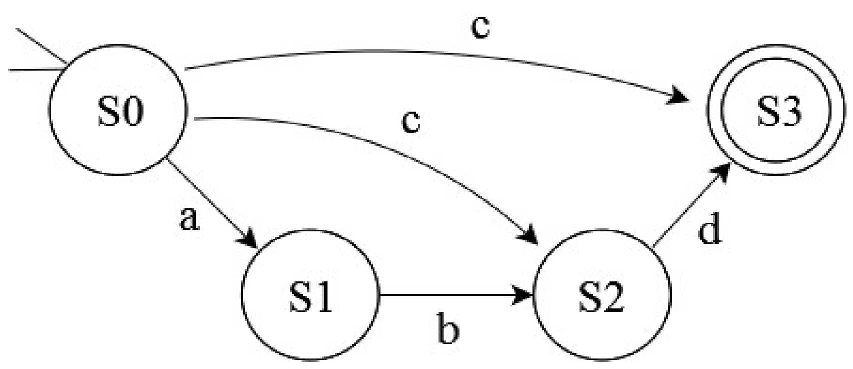
Name _________________________ NetID___________ Section_______

- **Do not open this exam** until everyone has an exam and the proctor tells you to begin.
- **Write clearly** – if we can't read or can't find your answer, it is wrong.
- There are 7 pages in this exam, including this one. Make sure you have them all.
- This exam is closed book – closed notes.
- You must be sure all electronic devices not explicitly exempted are:
    - silent
    - out of your view
    - off of your person
- Please be concise in your answers.

Question	Point Total	Scored Amount
I.	20	
II.	20	
III.	20	
IV.	20	
V.	20	
VI.	20	
VII.	20	
VIII.	20	
IX.	20	
Total:	200	

GL! HF!
|
8^D

I. Write a regular expression that matches a string of bits only if: it starts with a '0', ends with a '00', and has a '1' between every '010'.



II. Consider the following Finite State Automaton (FSA)

A. Is this a deterministic or a non-deterministic FSA? Why?

B. Write a regular expression that specifies the same language that this FSA accepts.

C. For each of the following strings, if it is accepted by the FSA above circle Yes and give a sequence of states that proves it is accepted. If not, circle No.

i. a — Yes: states ________________________ No

ii. a b d — Yes: states ________________________ No

iii. c d — Yes: states ________________________ No

iv. a b c d — Yes: states ________________________ No

v. c — Yes: states ________________________ No

D. Consider the language, “any number of '01's”.

Why can this language be represented by a FSA?

III. Consider the following grammar

(terminals are bolded and italicized, non-terminals are capitalized and underlined):

<u>LINE</u> → <u>PHRASE</u> | <u>PHRASE</u> ***@*** <u>LINE</u>
<u>PHRASE</u> → <u>WORD</u> | <u>PHRASE</u> ***&*** <u>WORD</u>
<u>WORD</u> → ***bar*** | ***baz*** | ***bo*** | ***boo*** | ***ba***

A. Draw a parse tree for the string baz @ boo & bo @ bar

B. The grammar implies a structure for the string below and that structure implies an order of operations. Add parentheses to the string to show this order of operations. Be sure to use enough parentheses to show the order unambiguously.

ba @ boo @ bo & bar & baz

IV. The scheme function (mult2-diff lst) should take one argument, a list of numbers, and results in multiplying each number in the list by two and then computing the difference of those numbers. e.g.: (mult2-diff '(9 1 2)) should evaluate to the difference of 18, 2 and 4: (18-2)-4, or 12. Finish mult2-diff below:

```
(define (mult2-diff lst)

      (if









        )
)
```

V. The scheme function (base2enc val) below takes a decimal number and encodes it in to a big-endian postive-magnitude-only bit value (i.e., each element of lst is either 0 or 1, with the lowest-ordered bit furthest to the right). It should represent the decimal number as a list of bits that make its binary equivalent. e.g.: (base2enc 15) results in 8+4+2+1, or '(1 1 1 1).
Base2enc does most of its computation using a helper funcion; (b2e lst accum num), which must use accumulator recursion and must be tail recursive. Finish the definitions below.
(Hint: the mod operator is your friend)

```
(define (base2end lst) (b2e num
                     _______________________  _____________  ______________

 ((define (b2e num accum lst)









)
```

VI. The scheme function checkbit takes two arguments: funcList, a list of functions on one parameter, and argList, a list of arguments corresponding to each function in funcList. The function checkbit evaluates to #t (true) if and only if all the functions in funcList resolve to a binary-coded list when run (i.e. a list that holds only '1's or '0's). You may presume argList holds the correct number of correct arguments.

Finish the definition of checkbit, below:

```
(define (checkbit funcList argList)









  )
```

VII. The Python function filter takes a function representing a boolean criteria and a series of values and returns a list holding on those values in the input list that make the boolean decision true. Write sifter, which applies the filter functionality to a list of lists, without using 'filter' or 'map', but using lambdas.

(Hint: It may help to think about how to write it in Scheme first and then modify the syntax)

Finish sifter below:

```
def sifter(criteria, megalist):
```

VIII. Suppose we have the following prolog program. Write in definitios for scaryc and scaryn such that scaryc will immediately fail the entire clause on any guardpet that isn't large, and that scaryn will fail only the scaryn subclause, but allow additional backtracking if a guardpet isn't large.

```
gp3(A):- domesticated(A), write([A, domesticated]), scaryc(A).  % uses scaryc, not scary
gp3(killerbee).

gp4(A):- domesticated(A), write([A, domesticated]), scaryn(A).  % uses scaryn, not scary
gp4(killerbee).

domesticated(elephant).
domesticated(horse).
domesticated(cat).
domesticated(goldfish).
domesticated(dog).

scary(A):- large(A), write([A, large]).
scary(A):- loud(A), write([A, loud]).

scaryc(A):-
scaryc(A):-

scaryn(A):-
scaryn(A):-

loud(dog).
loud(elephant).

large(elephant).
large(horse).
```

IX. For each of the following pairs of fact and goal, say whether they will unify in Prolog, and if so what bindings will be made.

	fact	goal	
A.	loo(B, B).	boo(X, X).	________________________
B.	boo(A, B, C).	boo(X, b, Z).	________________________
C.	noo([a, H | T]).	noo([x, Z]).	________________________
D.	goo(A, B).	hoo(a, b).	________________________
E.	hoo(x, y).	hoo(x, y).	________________________