

CMSC330 Spring 2008 Midterm #1

Discussion TA & Time (pick one): Eylul @ 10am Eylul @ 11am Wanli @ 11am

Do not start this exam until you are told to do so!

Instructions

- You have 50 minutes for to take this midterm.
- This exam has a total of 100 points, so allocate 30 seconds for each point.
- This is a closed book exam. No notes or other aids are allowed.
- If you have a question, please raise your hand and wait for the instructor.
- Answer essay questions concisely using 2-3 sentences. Longer answers are not necessary and a penalty may be applied.
- In order to be eligible for partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

1. (12 pts) Programming languages
 - a. Explain why goals for programming languages have changed since the 1960's
 - b. Do programs run faster when they are interpreted or compiled? Explain why.
 - c. Describe the syntax and semantics of a Ruby **while** loop. Make clear which is which.
 - d. Name a disadvantage of implicit variable declarations. Give a Ruby code example.
 - e. (4 pts) Explain why implicit variable declarations and static types are not considered orthogonal language features.

2. (18 pts) Ruby features

What is the output (if any) of the following Ruby programs? Write FAIL if code does not execute.

 - a. `3.times { |i| puts i }`
 - b. `x = 0`
`puts "x" if x`
`puts "not x" if !x`
 - c. `a = [1,2]`
`a[3] = "b"`
`a.push("a")`
`puts a`
 - d. `x = "car"`
`y = x`
`x = "cat"`
`puts y`
`puts "==" if x=="cat"`
`puts "equal" if !x.equal?("cat")`
 - e. `x = "CMSC 330"`
`x =~ /[([0-9])([0-9])([0-9])/`
`puts "#{ $1 } #{ $3 }"`
 - f. `h = { "guys" => 1, "gals" => 0 }`
`h["dudes"] = h["guys"] + 1`
`h.values.each { |x| puts x }`
 - g. `def down x`
 `until x < 0`
 `yield x`
 `x = x-1`
 `end`
`end`
`down(3) { |i| puts i }`
 - h. Describe the set of strings accepted by the Ruby regular expression `/^[^0-9]+/`
 - i. Describe the set of strings accepted by the Ruby regular expression `/^[0-9]+/`

3. (23 pts) Ruby programming

Consider the following programming problem. We need to read the transitions of an NFA from a text file and count how many transitions exist for each label, then print the labels in alphabetical order. A transition is a line of the form **fromState.label.toState**, where states and labels are separated by a period (.). State names are numbers (sequences of digits), and labels are either lowercase letters between a and z, or the letter E (for epsilon). Write a complete Ruby program that finds the name of the text file containing the transitions from the command line, opens the file, reads the transitions, counts the number of transitions for each label, then prints each label (in alphabetical order) and its # of transitions (separated by a space) on a separate line. Any line that does not contain a valid transition should be ignored.

<i>Example Input</i>	11.c.2	11.A.2
	1.a.22	21.b.9
	21.b.3	21a99
		3.E.45
		22.a.2
		22.b.2
<i>Example Output</i>	a 1	E 1
	b 1	a 1
	c 1	b 2

4. (10 pts) Regular expressions and languages

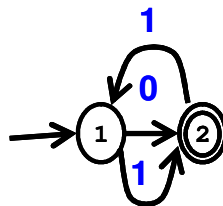
- Given the language $A = \{“aa”, ”c”\}$, what is the language A^1 ?
- Describe in English the language accepted by the regular expression $1(110)^*1$.
- Give a regular expression for all binary numbers that include more than two 0's.
- Give a regular expression for all binary numbers that don't include exactly two 0's.

5. (10 pts) Finite automata

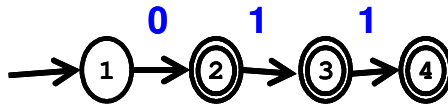
- (2 pts) How long could it take a DFA with n states and t transitions to accept a string with s symbols?
- (4 pts) For a given language, can the minimal size DFA required to accept the language be smaller or the minimal size NFA required? Explain why.
- (4 pts) The complement of a set S is the set of all elements not in S . Explain why for every regular expression R , we know there exists a regular expression \hat{R} that accepts only strings not accepted by R .

6. (27 pts) Finite automata

- (3 pts) Give a NFA for binary numbers with either an odd number of 0s or an odd number of 1s.
- (3 pts) Give a regular expression for the language accepted by the following DFA

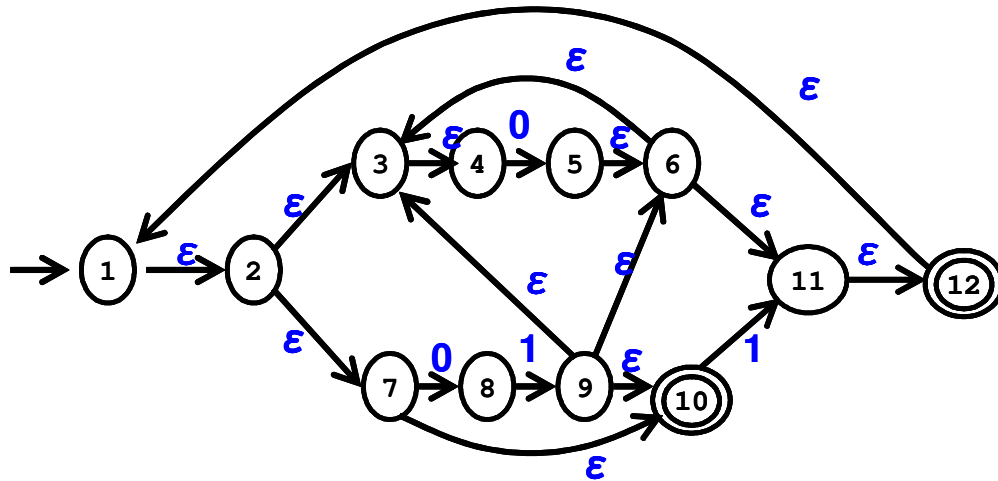


- c. (3 pts) Give the DFA that is the complement of the following DFA.



- d. (4 pts) Reduce the regular expression $1(0^* | 1)$ to an NFA, using the algorithm shown in class.

- e. (8 pts) Start reducing the following NFA to a DFA using the subset algorithm. Draw the start state of the DFA, and the states created upon transitions on 0 and 1. List the NFA states represented by each of your DFA states. Which of the DFA states are final states? (Note you only need to create 3 DFA states, not the full DFA)



- f. (6pts) Start applying Hopcroft reduction to minimize the following DFA. Show your initial partitions. Show the partitions resulting from your first split, and describe your reason for the split. (Show = list the DFA states in each partition)

