CMSC330 Spring 2013 Midterm #1

Name:

_____

Discussion Time	**9am**	**10am**	**11am**	**Noon**	**1pm**
TA Name (circle):	**Ilse**	**Daniel**	**Casey**	**Yoav**	**Ilse**
	Richard	**Richard**	**Richard**		

Instructions

- Do not start this test until you are told to do so!
- You have 75 minutes to take this midterm.
- This exam has a total of 100 points, so allocate 45 seconds for each point.
- This is a closed book exam. No notes or other aids are allowed.
- Answer essay questions concisely in 2-3 sentences. Longer answers are not needed.
- For partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

	Problem	Score
1	Programming languages	/6
2	Ruby	/6
3	Regular expressions & finite automata	/10
4	RE to NFA	/8
5	NFA to DFA	/16
6	DFA minimization	/10
7	Ruby programming	/28
8	OCaml	/16
	Total	/100

HONOR PLEDGE: I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/ examination. SIGNATURE: _____

1. (6 pts) Programming languages
   a. (3 pts) When would you use a scripting language like Ruby, instead of traditional language like Java or C? Briefly explain.

   b. (3 pts) Name one attribute of a good programming language that was discussed in lecture, and describe why it is desirable.

2. (6 pts) Ruby
   What is the output (if any) of the following Ruby programs? Write FAIL if code does not execute.

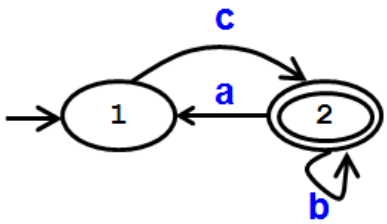
   a. (3 pts) **# Output =**

```
if "Sheldon" =~ /(S+[a-z][0-9]*)/
  puts $1
else
  puts "OP"
end
```

   b. (3 pts) **# Output =**

```
x = "Penny"
y = x
x = “Cent”
puts "50 #{y}"
```

3. (10 pts) Regular expressions and finite automata. Consider the following DFA:



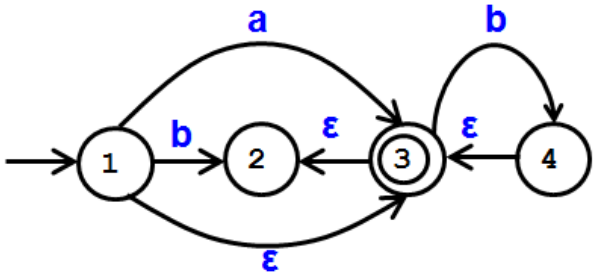
a. (4 pts) Give a regular expression for the strings accepted by the DFA. Use only the concatenate, union, and closure operations. I.e., do not use Ruby regular expressions.

RE =

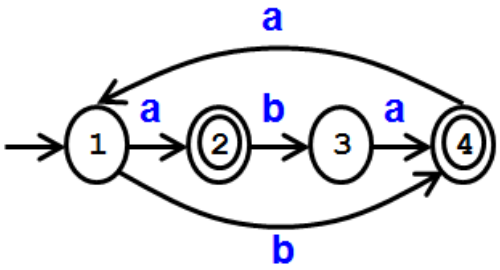
b. (6 pts) Give a DFA that accepts a string if and only if it is NOT accepted by the DFA above.

4. (8 pts) RE to NFA
Create a NFA for the regular expression **a|(bc)*** using the method described in lecture.

5. (16 pts) NFA to DFA
   Apply the subset construction algorithm discussed in class to convert the following NFA to a DFA. Show the NFA states associated with each state in your DFA.



6. (10 pts) DFA Minimization
Consider applying the Hopcroft DFA minimization algorithm discussed in class to the following DFA.



a. (2 pts) What are the initial partition(s) created by the Hopcroft algorithm?

b. (4 pts) Do any partitions need to be split? If yes, what is the result after splitting the partition?

c. (4 pts) Is the DFA minimization algorithm finished at this point? Explain.

7. (28 pts) Ruby programming

Implement a Ruby program that reads data from a file where each line has a name, followed by the amount (integer) a person spent on a particular item in a trip. Your program will read this file, compute the total amount a person spent, and display two lists. One list displays the total amount sorted by name and the second sorted by total amount. For example, running your program on a file called data.txt ("ruby bills.rb data.txt") will generate:

```
% more data.txt
John 7
Mary 200
Peter 14
Mary 6
Peter 3

%  ruby bills.rb data.txt
BY_NAME
John 7
Mary 206
Peter 17

BY_AMOUNT
206 Mary
17 Peter
7 John
```

You may assume names are lowercase and uppercase characters and that multiple spaces can appear between the name and the amount. Your program should output "BY_NAME", followed by the sorted list of names. It should then output "BY_AMOUNT", followed by a list sorted by amount. If a line has invalid data, your program will print the message "Wrong data detected" and will continue processing lines.

Helpful Functions	
f = File.new(n, mode)	// opens n in mode, returns File f
f.eof?	// is File object f at end?
ln = f.readline	// read single line from file f into String ln
a = f.readlines	// read all lines from file into array a
a = str.scan(…)	// finds patterns in String str, returns in array a
a = h.keys	// returns keys in hash h as an array a
a.sort	// returns sorted version of array a
a.sort!	// sorts elements of array a in place
a.size	// number of elements in the array
a.each { … }	// apply code block to each element in array
a.push / a.pop	// treat array as stack
ARGV	// array containing command line arguments

8. (16 pts) OCaml

   a. (2 pts each) Give the type of the following OCaml expressions.

      i. [["a"];["b";"c"]] **Type =**

      ii. let f y = (y + 1, 20) **Type =**

   b. (3 pts each) Give the value of the following OCaml expressions. If an error exists, describe it.

      iii. [7;8]::[9;10] **Value =**

      iv. let p q = (match q with h::t -> h) in (p [[10;20];[3;9]]) **Value =**

   c. (3 pts each) Write an OCaml expression with the following type.

      v. (float * float) list **Code =**

      vi. int list -> int **Code =**