CMSC330 Spring 2014 Midterm #1

Name:

_____

Discussion Time	**10am**	**11am**	**noon**	**1pm**	**2pm**
TA Name (circle):	**Tammy**	**Tammy**	**Tammy**	**Daniel**	**Daniel**
		Ilse	**Casey**	**Ian**	

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	Ruby	/9
2	Regular expressions & finite automata	/10
3	RE to NFA	/8
4	NFA to DFA	/16
5	DFA minimization	/9
6	Ruby programming	/30
7	OCaml	/18
	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. (9 pts) Ruby
   a. (3 pts) Is Ruby an imperative or functional programming language? Explain your answer.

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

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

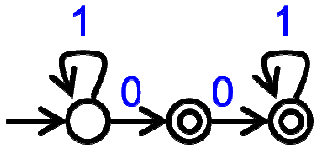
```
if "the compilnggg Game  " =~ /(p+ili?ng*) [a-zA-Z]+/
  puts $1
else
  puts "Game over"
end
```

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

```
b = []
b[1] = "mouse"
puts b[0] if b[1]
puts b[1]
```

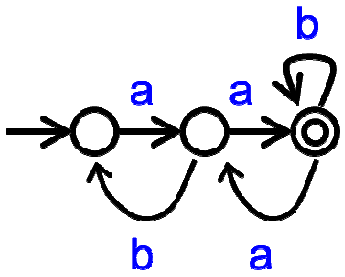
2. (10 pts) Regular expressions and finite automata.

   a. (4 pts) Consider the following DFA. 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.



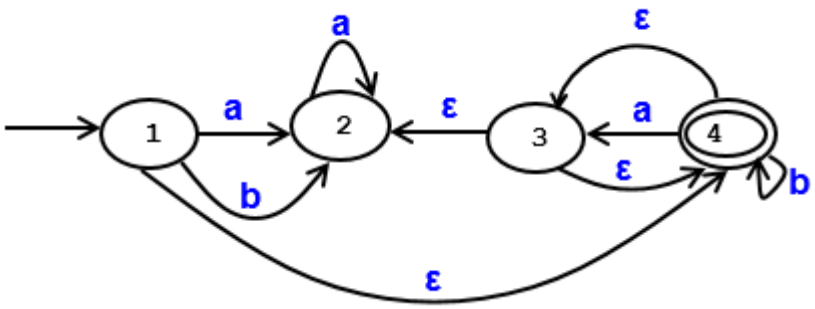
   RE =

b. (6 pts) Create a DFA that accepts a string (composed of a's and b's) if and only if it is NOT accepted by the following DFA.

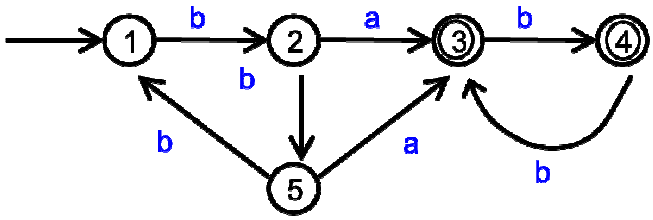


3. (8 pts) RE to NFA
Create a NFA for the regular expression **c(b|a*)** using the method described in lecture.

4. (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.



5. (9 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. (3 pts) Is the DFA minimization algorithm finished at this point? Explain.

6. (30 pts) Ruby programming

Implement a Ruby program that reads a course registration database file where each line contains a student name, course name and number of credits. Your program must process this file and display two lists. The first list displays each student followed by number of credits (**sorted** by student name). The second list displays each course followed by students taking the course separated by spaces (in any order). Your program will be called with the name of the database file as an argument. For example, running your program on a file called data.txt ("ruby roster.rb data.txt") could generate:

```
% more data.txt
Liz CMSC216 4
Liz CMSC250 3
Bob CMSC2 3
Bob CMSC216 4
% ruby roster.rb data.txt
ERROR Bob CMSC2 3
STUDENTS_INFO
Bob 4
Liz 7
COURSE_INFO
CMSC216 Liz Bob
CMSC250 Liz
%
```

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
s.to_i	// convert string s to int value
ARGV	// array containing command line arguments

Student names must be composed of one or more lowercase and uppercase characters. Course names must haves four uppercase letters followed by three digits. Course credit must be a single digit. Exactly one space separates each field. Lines that do not follow this format should produce an error message and otherwise be ignored.

While reading in the file, for each invalid line found, your program should output ERROR followed by the invalid line. Next, it should output "STUDENTS_INFO", followed by the list of students & credits (in sorted order by student name). Finally, it should output "COURSE_INFO", followed by the list of courses & students (the order courses or students are displayed does not matter).

7. (18 pts) OCaml

   a. (3 pts each) Give the type of the following OCaml expressions.

      i. [(2,3);(10,5)] **Type =**

      ii. fun x -> 4::x **Type =**

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

      iii. let f x = x+1 in f (f 2) **Value =**

      iv. let p q = (match q with x::y::t -> y) in (p [1;7;2;8]) **Value =**

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

      v. (string * int * string) list **Code =**

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