CMSC330 Spring 2013 Midterm #1 Solutions

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.

      Possible answers: text processing, to perform rapid prototyping

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

      Possible answers:
      - **Low cost of use**
      - **Portability**
      - **Programming environment - External support for the language, libraries, IDEs, etc.)**
      - **Clarify, simplicity and unity – Language provides framework for thinking about algorithms and a means of expressing those algorithms**
      - **Orthogonality – Every combination of features is meaningful; features work independently**
      - **Naturalness for the application – Program structure reflects logical structure of algorithm**
      - **Support for abstraction**
      - **Security and Safety, ease of program verification**

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

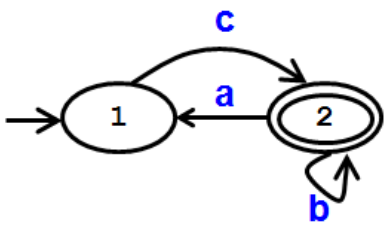
   a. (3 pts) **# Output = Sh**

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

   b. (3 pts) **# Output = 50 Penny**

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

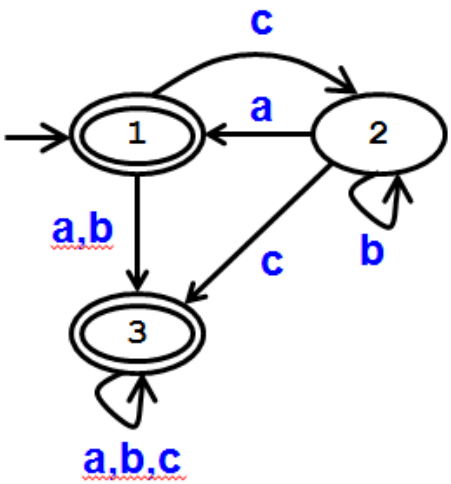
3. (10 pts) Regular expressions and finite automata. Consider the following DFA:



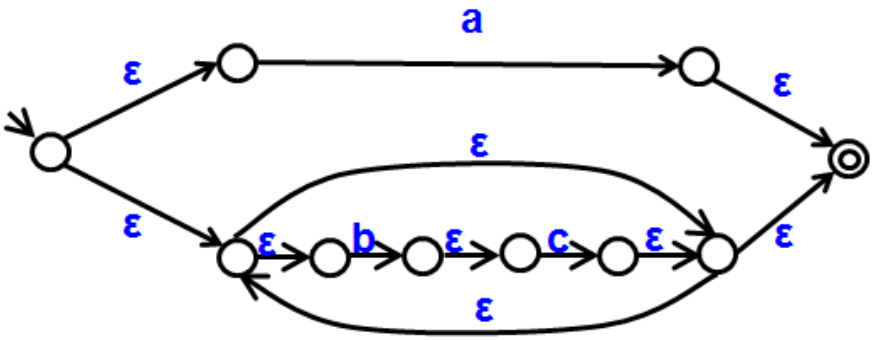
a. (4 pts) Give a regular expression for the strings accepted by the DFA. Use only the concatenate, union, and closure operations. I.e., do not use Ruby regular expressions.

RE = c(ac|b)*

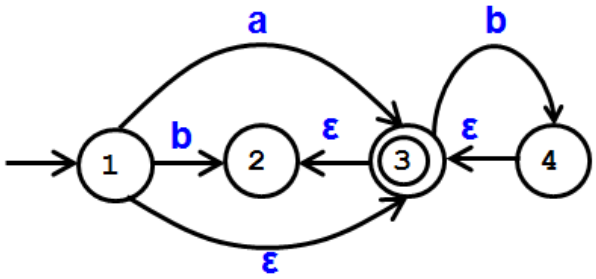
b. (6 pts) Give a DFA that accepts a string if and only if it is NOT accepted by the DFA above.



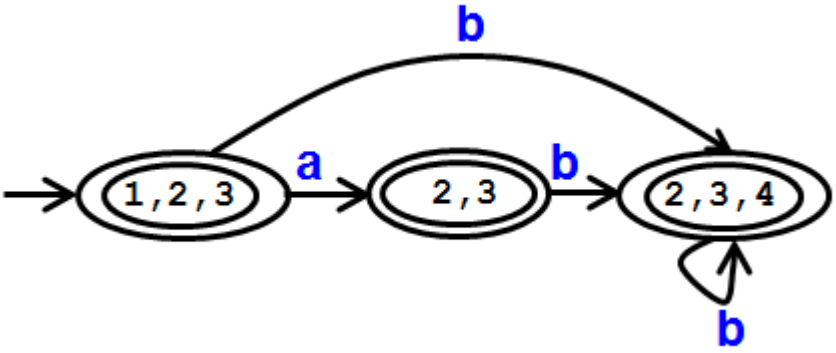
4. (8 pts) RE to NFA
Create a NFA for the regular expression **a|(bc)*** using the method described in lecture.



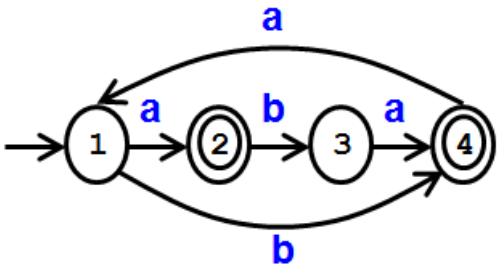
5. (16 pts) NFA to DFA
   Apply the subset construction algorithm discussed in class to convert the following NFA to a DFA. Show the NFA states associated with each state in your DFA.



Answer:



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?

Initial partitions = {1, 3} → P1
{2, 4} → P2

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

Both of them need to be split:

- **For P1:**
  move(1, b) →4 → P2
  move(3, b) → rejects

- **For P2:**
  move(2, a) → rejects
  move(4, a) → 1 → P1

  and

  move(2, b) → 3→P1
  move(4, b) → rejects

Result of split can be (one of the following) respectively

{1}, {3}, {2, 4}	**// Not yet minimal, since 2,4 can be split**
{1, 3}, {2}, {4}	**// Not yet minimal, since 1,3 can be split**
{1}, {2}, {3}, {4}	**// Minimal, since all states in own partition**

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

See comment above

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

Sample Answer #1:

```
f = File.new(ARGV[0], "r")
lines = f.readlines
bills = {}

lines.each { |line|
  if line =~ /^([a-zA-Z]+)\s+(\d+)$/
    bills[$1] = 0 if bills[$1] == nil
    bills[$1] += $2.to_i
  else
    puts "Wrong data detected"
  end
}

bills2 = bills.to_a

puts "BY_NAME"
bills2.sort! { |x,y| x[0] <=> y[0] }
bills2.each { |x| puts "#{x[0]} #{x[1]}" }
puts
puts "BY_AMOUNT"
bills2.sort! { |x,y| y[1] <=> x[1] }
bills2.each { |x| puts "#{x[1]} #{x[0]}" }
```

NOTE:

In place of bills2 = bills.to_a you can also explicitly convert the Hash to an Array of pairs using

```
bills2 = []
bills.each { |k,v| bills2.push([k,v]) }
```

or with

```
bills2 = []
bills.keys.each { |k| bills2.push([k,bills[k]]) }
```

Sample answer #2:

```
f = File.new(ARGV[0], "r")
lines = f.readlines
bills = {}

lines.each { |line|
  if line =~ /([a-zA-Z]+)\s+(\d+)/
    if bills[$1] == nil
      bills[$1] = $2.to_i
    else
      bills[$1] = bills[$1] + $2.to_i
    end
  else
    puts "Wrong data detected"
  end
}

puts "BY_NAME"
names_sorted = bills.keys.sort
names_sorted.each { |x|
  puts "#{x} #{bills[x]}"
}

puts "\nBY_AMOUNT"
amount_sorted = bills.sort{|a,b| -(a[1] <=> b[1])}
i = 0
while i < amount_sorted.length
  puts amount_sorted[i][1].to_s + " " + amount_sorted[i][0].to_s
  i += 1
end
```

8. (16 pts) OCaml

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

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

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

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

      iii. [7;8]::[9;10] **Value = ERROR, list is not int list list**

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

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

      v. (float * float) list **Code = [(1.3, 2.5); (6.7, 8.9)]**

      vi. int list -> int **Code = let f (x::y) -> x+1 OR**
      let f z = match z with (x::y) -> x+1