1. (4 pts) Ruby

Ruby has two classes Array and Hash that are similar but different in important ways. Both are frequently used to store a collection of data, and answer queries on the data (e.g., whether item x is present in the collection). Consider the difference between the following usages of Array and Hash in Ruby.

Given $a[x] = y$  // if a is an array, x is the **index**, y is the **element**  
// if a is a hash, x is the **key**, and y is the **value**

The method `include?`(x) is found in both the Array and Hash classes. For arrays, `include?`(x) returns true if x is one of the elements of the array. For hashes `include?`(x) returns if x is a key in the hash.

Consider the following code:

```ruby
a[2] = 3
x = a.include?(2)
y = a.include?(3)
```

a. (1 pt) If the code is preceded by the line `a = []`, what are the values of x & y?
   x = false, y = true

b. (1 pt) If the code is preceded by the line `a = {}`, what are the values of x & y?
   x = true, y = false

c. (2 pt) What is a simple alternative to writing `a.include?`(x) when “a” is a Hash? I.e., write code to test whether “x” is a key in Hash a without using “a.include?”. Use “if a[x] != nil” instead of “a.include?“(x)”

2. (6 pts) RE to NFA

Create a NFA for $c(a|b)^*$, using the algorithm discussed in class.
(10 pts) Consider the following NFA.

a. (2 pts) Does the NFA accept the string “bba”? If it accepts the string, list a sequence of state transitions (e.g., 1,2,3) that leads to acceptance of “bba”.

Yes. 1,2,3,1,3.

b. (8 pts) Convert the NFA to a DFA using the subset construction algorithm discussed in class. Be sure to label each state in the DFA with the corresponding state(s) in the NFA.