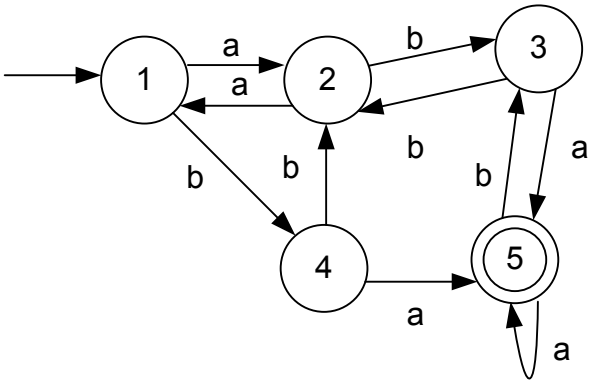
CS 315-Theory of Automata (Spring 2015)
Assignment 2

Due Date: 04-03-2016 **Time: 11:55 pm** **Where: Submit the attachment on LMS** **Marks: 110**

Problem 1 **(15 points)**

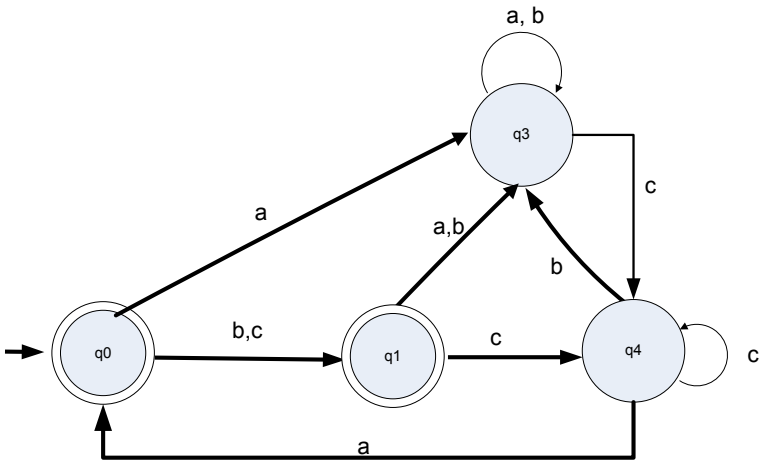
Find the minimal DFA (if any) for the following DFA by filling the table given below. After filling the table how you can decide about the equivalence of states and then draw the DFA if required.



2				
3				
4				
5				
	1	2	3	4

Problem 2 **(15 points)**

Convert the following DFA in to the regular expression.



Problem 3 **(20 points)**

Convert the following RE into the NFA and the DFA.

$$\mathbf{Re = a^{+}b(c|d)e^{*}|cb^{+}\ (b|e|c)^{*}}$$

Coding Exercises:

* Please use the "re" library in python for this assignment.
Here is sample python code for a jumpstart. It finds names and ages in a string using regular expression

```
import re

inputStr = '''Ali is 81 years old and Ayesha is 42 '''

ages = re.findall(r'[0-9]+ ',inputStr)
names = re.findall(r'[A-Z][a-z]*',inputStr)

print(ages)
print(names)
```

Output is

['81', '42']
['Ali', 'Ayesha']

Problem 4 **(10 points)**

Write a program to find the sum of all the digits in a string (using regular expression)
Example:

Input:
inputStr ='''' how 4 are 6 you 7? '''

Output:
Sum is 17

Problem 5 **(20 points)**

Write a function to remove all the dead states from deterministic finite automata. You must traverse the all states of machine and find the dead states (a state with no outgoing transition or just a self loop) except final states then remove them from the given dictionary and return the updated dictionary.

Input:

edgesMap: {
(0:'a'): [1]
(1:'a'): [2]
(2:'a'): [3]
(3:'a'): [3]
(3:'b'): [3]
(0:'b'): [4]
(1:'b'): [4]
(2:'b'): [4]
(4:'b'): [4]
}
acceptingStates = [3]

Output:

edgesMap: {
(0:'a'): [1]
(1:'a'): [2]
(2:'a'): [3]
(3:'a'): [3]
(3:'b'): [3]
}

Problem 6 **(10 points)**

Write a functions which returns true for all the string having **at most one** '+' sign between two words and false for rest of the strings (the only character that could be between words is '+' so you do not need to handle exceptions). Make only one regular expression to accommodate all conditions

Input:
1- "ali+ahmed"
2- "hello+world+again"

Output:
1- true
2- false

Problem 7 **(20 points)**

We have covered NFA simulator in class now write a function to simulate the NFA even with empty transitions represented by **E** in dictionary (You are not allowed to edit the input string except truncating it)

simulateNFA (input, currState, edgesMap, acceptingStates)

Input: input string
currState: Current state(Starting state in case of first ietration)
edgesMap: Dictionary of the state machine
acceptingStates: Set of accepting states of machine

Here is a sample edgesMap to test your code with

edgesMap: {
(0:'E'): [1]
(1:'a'): [1]
(1:'b'): [2]
(2:'a'): [1]
(2:'b'): [2]
(2:'E'): [3]

}

acceptingStates = [2]