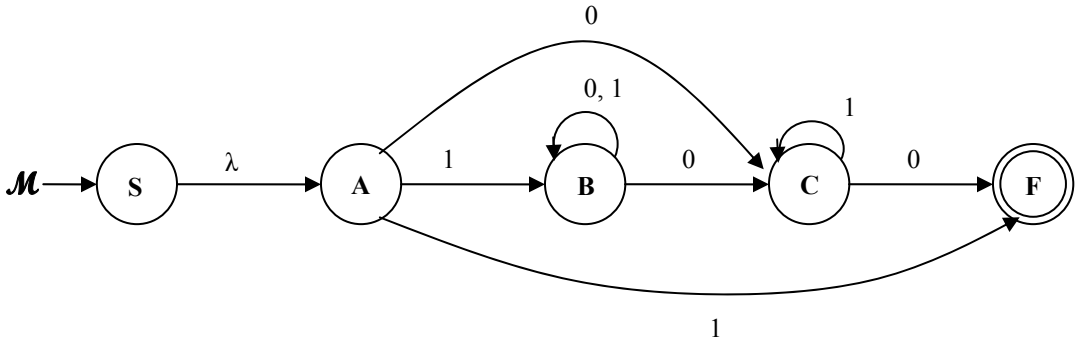
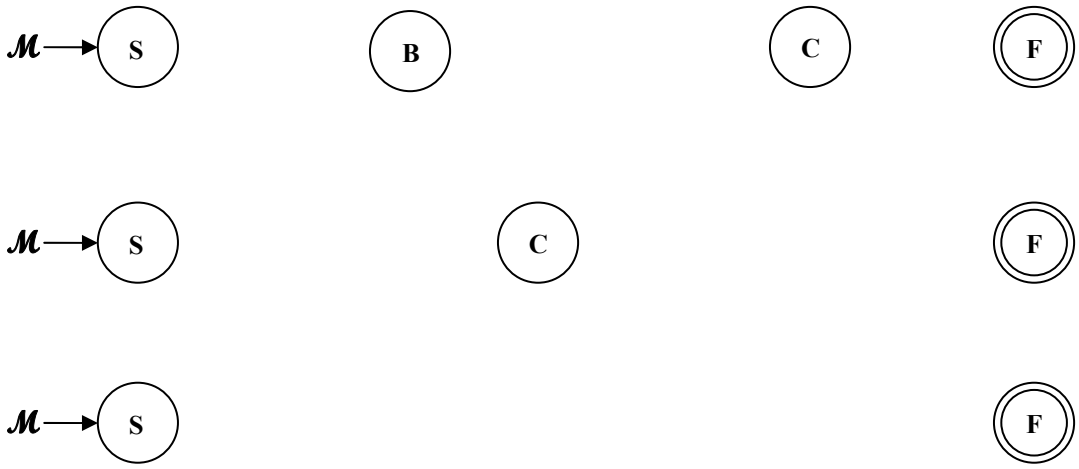
COT 5310 Fall 2006 Review Assignment #1 Name: *KEY*

4 **1.** Present a Mealy Model finite state machine that reads an input **x** $\in$ **{0, 1}*** and produces the binary number that represents the result of incrementing **x** by 1 (assumes all numbers are positive, including results). Assume that **x** is read starting with its least significant digit.
Examples: **0010 → 0011; 1011 → 1100; 1111 → 0000** (wrong answer due to overflow)

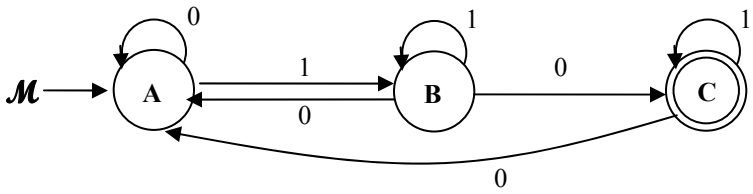
6 **2.** Let **L** be defined as the language accepted by the finite state automaton $\mathcal{M}$:



Using the technique of collapsing states, replacing transition letters by regular expressions, develop the regular expression associated with a that generates **L**. I have included the diagrams associated with removing states **A**, **B**, then **C**, in that order. You must use this approach of collapsing one state at a time, showing the resulting regular expressions.



3. Let **L** be defined as the language accepted by the finite state automaton $\mathcal{M}$:



6 **a.)** Present the regular equations associated with each of $\mathcal{M}$'s states, solving for the language recognized by $\mathcal{M}$.

b.) Assuming that we designate $\mathcal{a}$ as state **1**, **B** as state **2** and **C** as state **3**. Kleene's Theorem allows us to associate regular expressions $R_{i,j}^{k}$ with $\mathcal{M}$ *where* $R_{i,j}^{k}$ is the set of strings formed by going from state **i** to State **j** without passing through states whose indices are higher than **k**.

4 Expressed in terms of these $R_{i,j}^{k}$, what is the language recognized by $\mathcal{M}$?

$L(\mathcal{a}) = R_{1,3}^{3}$

4 **c.)** Present a regular grammar that generates **L**, the language recognized by $\mathcal{M}$.

4. Consider the regular grammar **G**:

S → 0 S | 1 A | 0 A

A → 1 S | 0 A | 0 B | λ

B → 0 S | 0 B

4 **a.)** Present an automaton $\mathcal{M}$ that accepts the language generated by **G**:

b.) Regular grammars generate the class of regular languages. Regular expressions denote the class of regular sets. The equivalence of these is seen by a proof that every regular set is a regular language and vice versa. The first part of this, that every regular set is a regular language, can be done by first showing that the basis regular sets (**Ø , { λ} , { a | a ∈ Σ }**) are each generated by a regular grammar over the alphabet $\sum$.

3 **i.)** Demonstrate a regular grammar for each of the basis regular sets. The empty set has been done for you.

Ø **G = { {S}, Σ, S, {S → S}}**

{ λ}

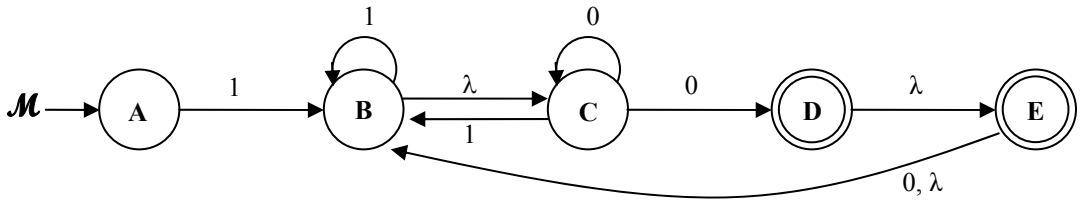
{ a }

Let **L** be generated by the regular grammar **G = (N , Σ , S , P)**.

4 **ii.)** Present a construction that produces a regular grammar for **L***

2 **iii.)** This shows that regular grammars are closed under Kleene *. What remains to be done to show that every regular set is a regular language? Don't do the proof, just state the steps that need to be carried out.

5. Let **L** be defined as the language accepted by the finite state automaton $\mathcal{M}$:



2 **a.)** Fill in the following table, showing the λ-closures for each of **A**'s states.

State	A	B	C	D	E
λ-closure					

4 **b.)** Convert **M** to an equivalent deterministic finite state automaton. Use states like **AC** to denote the subset of states **{A,C}**. Be careful -- λ-closures are important.

2 **c.)** What is a simple regular expression for the language recognized by $\mathcal{M}$. This should be easy to see from the above DFA.

3 **6.** Show the following grammar is ambiguous by presenting two distinct leftmost derivations of some string in the associated language. You must either show every step of the two leftmost derivations from **S** to your chosen string, or present two distinct parse trees for this chosen string.

S → #S | S# | A **A → #A# | a** **N = {S,A}, ∑ = {#,a}**

7. Let **L** be recognized by the DFA, $\mathcal{M}=(Q , \Sigma , \delta, q_0 , F)$, where $|Q|=N$.

3 **a.)** State the strong Pumping Lemma for **L**, in terms of **N**. Be precise and complete.

5 **b.** Use the strong Pumping Lemma for regular languages to show that the following language, $L = \{ a^n \mid n \text{ is prime} \}$, is not regular.

7 **8.** Let $R = Q + RP$, where $\lambda \notin P$. Prove every solution for **R** is contained in QP^*.
Hint: Consider an arbitrary $w \in R$. Show $w \in QP^*$. Base proof on length of **w**.

10 **9.** Given a finite state automaton denoted by the transition table shown below, and assuming that **6** is the sole final states, fill in the equivalent states matrix I have provided. Use this to create an equivalent, minimal state automaton.
Note: In a moment of uncharacteristic kindness I filled in the easy parts -- final cannot merge with non-final

	a	b	c
1	4	3	2
2	2	1	5
3	4	1	4
4	4	1	5
5	2	4	6
6	4	2	5

2					
3					
4					
5					
6	X	X	X	X	X
	1	2	3	4	5

Don't forget to construct and write down your new, equivalent automaton!!

7 **10.** Consider the pushdown automaton $\mathcal{A}$ = **({ q, f } , { a , b , c } , { A, Z }, δ, q, Z, { f })** , where δ defines transitions:

δ(q , a , Z) = **{ (q , A) }**
δ(q , b , A) = **{ (q , A A) }**
δ(q , ç , A) = **{ (f , λ) }**
δ(f , a , A) = **{ (f , A) }**
δ(f , b , A) = **{ (q , A A) }**
δ(f , c , A) = **{ (f , λ) }**

Write the equivalent grammar.
Hint: the starting non-terminal is: **< q , Z , f >**, meaning generate all string that are consumed when we start in **q**, and end up in **f**, having uncovered what's below **Z**.

G = { {[q'xq''] | q', q''∈ {q,f} and x ∈ {A,Z}}, {a,b,c}, P, S }

P =
{ S → [qZf] } ∪
{ [qAf] → c, [fAf] → c } ∪
{
[qZf] → a[qAf],
[qAf] → b[qAq][qAf] | b[qAf][fAf],
[qAq] → b[qAq][qAq] | b[qAf][fAq] ,
[fAf] → a[fAf],
[fAf] → b[qAq][qAf] | b[qAf][fAf],
[fAq] → a[fAq],
[fAq] → b[qAq][qAq] | b[qAf][fAq] }

[qZq], [fZq], and **[fZf]** never arise and so do not need to be specified.

11. Consider some language **L**. For each of the following cases, write in one of **(i)** through **(vi)**, to indicate what you can say conclusively about **L**'s complexity, where

(i) **L** is definitely regular
(ii) **L** is context-free, possibly not regular, but then again it might be regular
(iii) **L** is context-free, and definitely not regular
(iv) **L** might not even be context-free, but then again it might even be regular
(v) **L** is definitely not regular, and it may or may not be context-free
(vi) **L** definitely is not even context-free

Follow each answer with example languages **A** (and **B**, where appropriate) to back up the complexity claims inherent in your answer; and/or state some known closure property that reflects a bound on the complexity of **L**.

Example.) $\mathbf{L = A \cup B}$, where **A** and **B** are both context free, and definitely not regular

L can be characterized by **Property (ii)**, above.
L is context-free, since the class of context-free languages is closed under union.
L can be regular. For example,
$\mathbf{A = \{ a^n b^m \mid m \geq n \}}$, $\mathbf{B = \{ a^n b^m \mid m \leq n \}}$,
$\mathbf{L = A \cup B = \{ a^n b^m \mid n, m \geq 0 \}}$, which is regular since it can be represented by the regular expression **a*b***.
But **L** is in general not guaranteed to be regular, e.g., if we just make **A** and **B** the same context-free, non-regular set, then $\mathbf{L = A \cup A = A}$, which is not regular.

4 **a.)** $\mathbf{L = A^*}$ and $\mathbf{\Sigma \subseteq A}$

4 **b.)** $\mathbf{L = A - B}$, where **A** is context-free, not regular, and **B** is regular

4 **c.)** $\mathbf{L = \{ww^R \mid w}$ is in **A** and **A** is regular$\}$

4 **d.)** $\mathbf{A = L - B}$ where **B** is regular and **A** is not context free

4 **e.)** $\mathbf{A = L \cup B}$ where **B** is context free but not regular

4 **f.)** $\mathbf{L \subset A}$, where **A** is regular

4 **g.)** Let $\boldsymbol{\sigma}$ **b**e a mapping from $\boldsymbol{\Sigma}$ into CFGs, such that $\boldsymbol{\sigma(a) = g_a}$, for some CFG $\mathbf{g_a}$. Let **A** be a context free, non-regular language defined by $\mathbf{A = \sigma(L)}$.

6 **12.** Analyze the following language, **L**, proving it non-regular by showing that there are an infinite number of equivalence classes formed by the relation $\mathbf{R_L}$ defined by:

$$\mathbf{x}\ \mathbf{R_L}\ \mathbf{y} \text{ if and only if } [\ \forall \mathbf{z} \in \mathbf{\{a,b,c\}^*},\ \mathbf{xz} \in \mathbf{L} \text{ exactly when } \mathbf{yz} \in \mathbf{L}\ ].$$

where $\mathbf{L = \{\ a^n\ b^m\ c^t \mid t = m + n\ \}}$.

You don't have to present all equivalence classes, but you must demonstrate a pattern that gives rise to an infinite number of classes, along with evidence that these classes are distinct from one another.

6 **13.** Use the Pumping Lemma or Ogden's Lemma for context-free languages to show that **L** is not context-free. Again, $\mathbf{L = \{\ a^n\ b^m\ c^t \mid t = m + n\ \}}$.

6 **14.** Consider the language **L = { $a^n b^m c^t$ | t = n or t = m }**. Present a context-free grammar that produces this language.

4 **15.** Again, consider the context-free language **L = { $a^n b^m c^t$ | t = n or t = m }**. What language results when we take the **Max** of this language? What about the **Min**?

Max(L) = { w | w ∈ L, and if **wy ∈ L**, then **y = λ }**

Min(L) = { w | w ∈ L, and if **xy = w** and **x ∈ L**, then **y = λ }**

6 **16.** Consider the GNF CFG **G = ({ S, T, C, D } , { a , b, c, d } , S , P)** where **P** is:

S → a S T | a D D **C → c**

T → b D T | b S C **D → d**

Present a pushdown automaton that accepts the language generated by this grammar. Your PDA must accept by empty store, it must start with **S** on its stack, and it must be based on the above grammar (hint: only one state is needed).

8 **17.** Consider the context-free grammar **G = ({ S } , { a , b } , S , P)**, where **P** is:

S → S a S b S b S | S b S a S b S | S b S b S a S | λ

Provide the first part of the proof that

L (G) = L = { w | w has twice as many **b**'s as **a**'s **}**

That is, show that **L (G) ⊂ L**

To attack this problem we can first introduce the notation that, for a syntactic form α, α_a = the number of **a**'s in α, and α_b = the number of **b**'s in α.

Using this, we show that if $S \overset{+}{\Rightarrow} \alpha$, then $2\alpha_a = \alpha_b$ and hence that **L (G) ⊂ L:**

A straightforward approach is to show, inductively on the number of steps, **i**, in a derivation, that,

if $S \overset{i}{\Rightarrow} \alpha$, then $2\alpha_a = \alpha_b$

Basis (i = 1):

IH:

IS:

18. Consider the context-free grammar **G = ({ S , E, L, R } , { a , –, <comma>} , S , P)**, where **P** is:
S → E | E, S
E → L – R
L → a
R → L | LR

3 **a.)** Remove all chain (unit) rules from **G**, creating an equivalent grammar **G**'.

5 **b.)** Convert grammar **G'** to its Chomsky Normal Form equivalent, **G''**. In CNF, each production has the form **A → B C** or **A → a** where **B** and **C** are non-terminals and **a** is a terminal.

6 **c.)** Convert grammar **G''** to its Greibach Normal Form equivalent, **G'''**. In GNF, each production has the form $\mathbf{A \rightarrow a\ A_1\ A_2 \ldots\ A_k}$ where **a** is terminal and $\mathbf{A_i}$ is non-terminal for $0 \leq i \leq k$ or $\mathbf{S \rightarrow \lambda}$.

7 **19.** Present the **CKY** recognition matrix for the string **b b a a b a** assuming the grammar specified by the rules

S → L A | M B | A A | B B
L → A S
A → a
M → B S
B → b

The first iteration has been filled in.

	b	**b**	**a**	**a**	**b**	**a**
1	**{B}**	**{B}**	**{A}**	**{A}**	**{B}**	**{A}**
2						
3						
4						
5						
6						