Written Assignment 1
Due ???

This assignment asks you to prepare written answers to questions on regular languages and finite automata. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work. Remember that written assignments are to be turned in either at the start of lecture or in the CS164 homework box in 283 Soda by 12:30 PM on the due date.

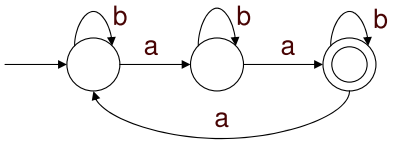
1. Consider the following languages over the alphabet $\Sigma = \{a, b\}$.

   - L_1: *All strings that contain at least two a's*
   - L_2: *All strings that contain at least one b*
   - L_3: *All strings that contain at least two a's and at least one b*
   - L_4: *All strings that contain at most one a or no b's*

   Give a deterministic finite automaton (DFA) for the languages L_1, L_2, L_3 and L_4.

   Aside: This example illustrates that regular languages are closed under intersection and complementation. Note that $L_3 = L_1 \cap L_2$ and $L_4 = \Sigma^* - L_3$, where Σ^* represents the language containing all strings over the alphabet Σ.

2. Consider the following DFA over the alphabet $\Sigma = \{a, b\}$.

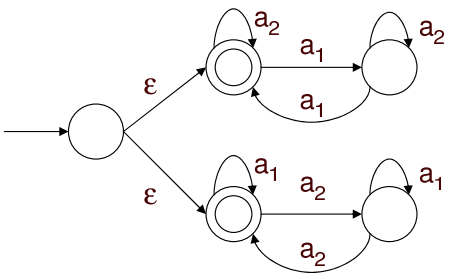
   

   Give a one-sentence description of the language recognized by the DFA. Write a regular expression for the same language.

3. Let $\Sigma_m = \{a_1, \ldots, a_m\}$ be an alphabet containing m elements, for some integer $m \geq 1$. Let L_m be the following language that includes all strings in which at least one of the characters occurs an even number of times, i.e.

   All strings in which a_i occurs an even number of times for some i, where $1 \leq i \leq m$

   The following figure shows an NFA for the language L_2.

   Construct a DFA for the language L_2. Also construct an NFA for the language L_3.

Aside: Non-deterministic finite automata (NFAs) are no more powerful than DFAs in terms of the languages that they can describe. However, NFAs can be exponentially more succinct than DFAs, as this problem demonstrates. For the language L_m, there exists an NFA of size at most $2m + 1$ while any DFA must have size at least 2^m. Note that the DFA for the language L_3 is not as easy to construct as the NFA for the language L_3.

4. (a) Determine whether or not the following languages are regular. Explain why in one or two sentences.
   - L_1: *All strings over the alphabet* $\{0, 1\}$ *that have equal number of 1's and 0's.*
   - L_2: *All strings over the alphabet* $\{0, 1\}$ *that are palindromes* [1].
   - L_3: *All words in the Oxford English dictionary.*

   (b) The Cool language as described on page 16 of the Cool reference manual is not regular (The alphabet here is the set of all tokens, and the language is the set of all valid Cool programs). Give one reason why.

   Aside: This illustrates that we cannot use a lexer to *parse* the Cool language.

[1] strings that read the same when read left to right or right to left

Written Assignment 2
Due ??

This assignment asks you to prepare written answers to questions on context-free grammars. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

1. Let L be the language consisting of all palindromes over the alphabet $\Sigma = \{a, b\}$. That is, L consists of all sequences of a's and b's that read the same forward or backward. For example, $aba \in L$ and $aabbbaa \in L$, but $abb \notin L$.

   Write a context-free grammar for the language L.

2. Consider the following grammar:

$$\begin{aligned} S &\rightarrow aSb \\ S &\rightarrow aS \\ S &\rightarrow \epsilon \end{aligned}$$

   (a) Give a one-sentence description of the language generated by this grammar.

   (b) Show that this grammar is ambiguous by giving a string that can be parsed in two different ways. Draw both parse trees.

   (c) Give an unambiguous grammar that accepts the same language as the grammar above.

3. Using the context-free grammar for Cool given in Section 11 of the Cool manual, draw a parse tree for the following expression.

```
if not x = 0 then
    y <- z + 2 * x + 1
else
    y <- z <- 0
fi
```

   Note that the context-free grammar by itself is ambiguous, so you will need to use the precedence rules in Section 11.1 to get the correct tree.

4. Give an example of a grammar that is $LL(2)$ but not $LL(1)$.

Written Assignment 3
Due ???

This assignment asks you to prepare written answers to questions on LL and LR parsers. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

1. Use left-factoring and/or elimination of left recursion to convert the following grammars into LL(1) grammars. You may assume that these grammars are unambiguous.

   (a)

   $$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow int \mid (E) \end{aligned}$$

   (b)

   $$L \rightarrow int \mid int\ ,\ L \mid (L)$$

   (c)

   $$A \rightarrow int \mid int + A \mid int - A \mid A - (A)$$

2. Consider the following grammar describing a certain sort of nested lists:

   $$\begin{aligned} S &\rightarrow T\ ;\ S \mid \epsilon \\ T &\rightarrow U \bullet T \mid U \\ U &\rightarrow x \mid y \mid [S] \end{aligned}$$

   The nonterminals are S, T, and U, while the terminals are ;, $\bullet$, [,], x, and y.

   (a) Left-factor this grammar.

   (b) Give the First and Follow sets for each nonterminal in the grammar obtained in part (a).

   (c) Using this information, construct an LL parsing table for the grammar obtained in part (a).

   (d) Suppose we generated an LL parser for the grammar using the table you constructed. What would go wrong if it tried to parse the following input string?

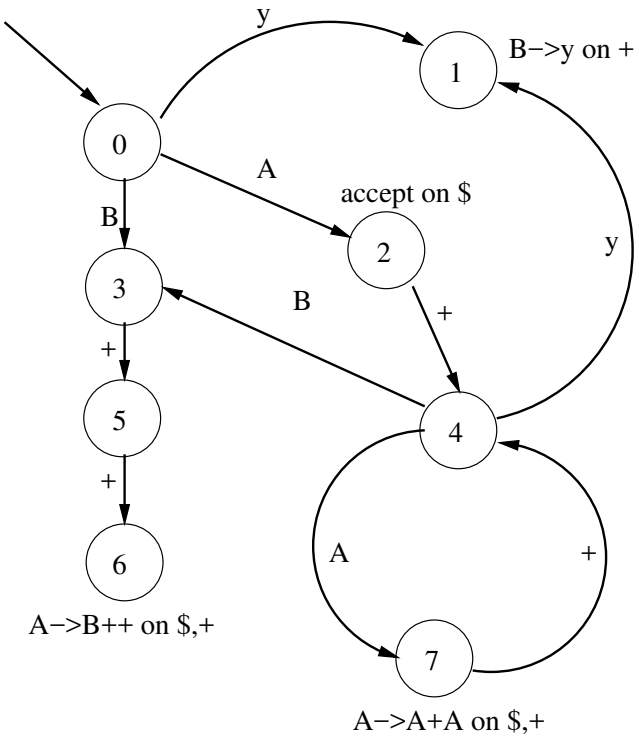
   $$[\ x\ ;\ y\ ]\ \bullet\ [\ ;$$

   (That is, when we get an error, how much of the input string has been consumed, and what is the parser trying to do?)

3. Consider the following LR(1) grammar:

$$
\begin{array}{lcll}
S & \rightarrow & A & \\
A & \rightarrow & A + A \mid B + + & \text{(Each '+' is a separate token.)} \\
B & \rightarrow & y &
\end{array}
$$

and its corresponding DFA:



Complete the table below, showing the trace of an LR(1) parser (which uses the DFA above) on the input provided. The "Stack" column must show the stack (with the top at right), the "Input" column shows the not-yet-processed input terminals, and the "Action" column must show whether the parser performs a shift action or a reduce action or accepts the input. In the case of a reduce action, please indicate which production is used.

Stack (with top at right)	Input	Action
	▶ y + + + y + + $	shift
y		

Written Assignment 4
Due ??

This assignment asks you to prepare written answers to questions on LR parsers. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

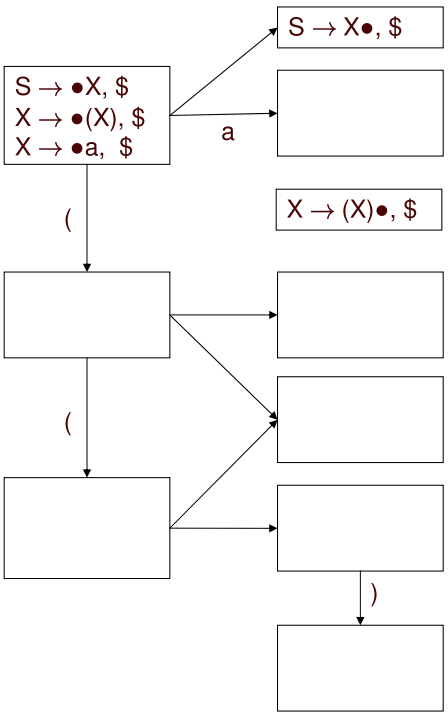
1. In each of the following cases, give as simple a grammar as you can.
   (a) Give an LR(1) grammar that is not LL(1). Explain why your grammar is LR(1) and not LL(1).
   (b) Give an unambiguous grammar that is not LR(1). Explain why your grammar is unambiguous and not LR(1).

   Hint: In each of the above cases, there exists a grammar that generates a language with only two strings.

2. Consider the following grammar with start symbol S:

$$\begin{array}{lcl} S & \rightarrow & X \\ X & \rightarrow & (X) \mid a \end{array}$$

The following figure shows a skeleton of the LR(1) parsing DFA for this grammar.



   (a) Complete the DFA skeleton. You need to fill in the LR(1) items for each state of the DFA, add new transitions, and label each transition. (You should not add any new states though.)
   Hint: One of the states has a self-loop.
   (b) Is the grammar LR(1)? Is the grammar LR(2)? Is the grammar LL(1)? Why or why not?
   (c) Use your DFA to parse $((((a))))$. Show the sequence of shift/reduce steps.

Written Assignment 1
Due ??

This assignment asks you to prepare written answers to questions on regular languages and finite automata. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together, however, your write-up must be your own individual work.

1. Give a DFA for each of the following languages over the alphabet $\Sigma = \{a, b\}$.

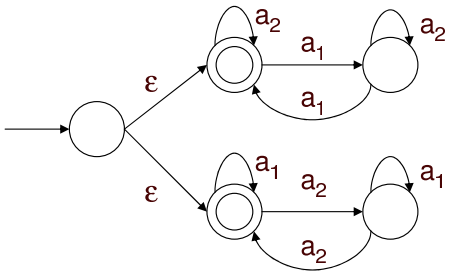
   - L_1: *All strings that contain at least two a's*
   - L_2: *All strings that contain at least one b*
   - L_3: *All strings that contain at least two a's and at least one b*
   - L_4: *All strings that contain at most one a or no b's*

   Aside: This example illustrates that regular languages are closed under intersection and complementation. Note that $L_3 = L_1 \cap L_2$ and $L_4 = \Sigma^* - L_3$, where Σ^* represents the language containing all strings over the alphabet Σ.

2. Let $\Sigma_m = \{a_1, \ldots, a_m\}$ be an alphabet containing m elements, for some integer $m \geq 1$. Let L_m be the following language that includes all strings in which at least one of the characters occurs an even number of times, i.e.

   All strings in which a_i occurs an even number of times for some i, where $1 \leq i \leq m$

   The following figure shows an NFA for the language L_2.



   Construct a DFA for the language L_2. Also construct an NFA for the language L_3.

   Aside: Non-deterministic finite automata (NFAs) are no more powerful than DFAs in terms of the languages that they can describe. However, NFAs can be exponentially more succinct than DFAs, as this problem demonstrates. For the language L_m, there exists an NFA of size at most $2m+1$ while any DFA must have size at least 2^m. Note that the DFA for the language L_3 is not as easy to construct as the NFA for the language L_3.

3. Write regular expressions for the following languages over the alphabet $\Sigma = \{0, 1\}$:

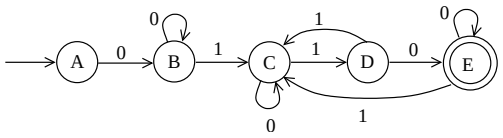
   (a) All strings that contain at least one 0 and at least one 1 and that also end with at least two 1s.

   (b) All strings that do not begin with 01.

   (c) All strings that contain an odd number of 1s.

   (d) All strings that contain exactly two 1s and at least one 0.

4. Give a one-sentence description and a regular expression for the language over the alphabet $\Sigma = \{0, 1\}$ described by the following deterministic finite automaton (DFA):



5. For each of the following lexical specifications, give a regular expression describing the language of possible outputs. Assume that all inputs are strings of 0s and 1s only.

   (a) Specification 1:

```
0       { print ("c"); }
00      { print ("a"); }
1       { print ("b"); }
```

   (b) Specification 2:

```
(0+)(1+) { print("a"); }
0         { print("b"); }
1         { print("c"); }
```

Written Assignment 2
Due ??

This assignment asks you to prepare written answers to questions on context-free grammars, parse trees, and parsing. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together, but your write-up must be your own individual work.

1. Give a context-free grammar (CFG) for each of the following languages over the alphabet $\Sigma = \{0, 1\}$:
   (a) All nonempty strings that start and end with the same symbol.
   (b) All strings that contain more 1s than 0s.
   (c) All palindromes (a palindrome is a string that reads the same forwards and backwards).

2. Consider the following grammar:

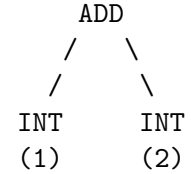
$$\begin{aligned} S &\rightarrow aSb \\ S &\rightarrow aS \\ S &\rightarrow \epsilon \end{aligned}$$

   (a) Give a one-sentence description of the language generated by this grammar.
   (b) Show that this grammar is ambiguous by giving a string that can be parsed in two different ways. Draw both parse trees.
   (c) Give an unambiguous grammar that accepts the same language as the grammar above.

3. The Cool Reference Manual, in Chapter 11, has the context-free grammar defining the syntax for Cool. Create the parse tree from the grammar definition for the following class definition.

```
class FOO inherits BAR {
  i : Int <- 42;
  baz (x:Int) : Foo { i <- x + i };
};
```

   You do not need to construct nonterminals that immediately produce terminals, or include terminals which add no meaning because they are captured by the nature of the production. For example, for the expression "1 + 2", the following parse tree would be sufficient:



4. Give a one-sentence description of the language generated by each of the following CFGs.

(a)

$$\begin{aligned} S &\rightarrow Z1Z1Z1A \\ Z &\rightarrow Z0 \mid \varepsilon \\ A &\rightarrow A0 \mid A1 \mid \varepsilon \end{aligned}$$

(b)

$$S \rightarrow 0 \mid 1 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1$$

5. Give an example of a simple grammar that is $LL(2)$ but not $LL(1)$.

6. Consider the following CFG, which has the set of terminals $T = \{\mathbf{id}, (,), [,], ;\}$.

$$\begin{aligned} E &\rightarrow \mathbf{id} \mid \mathbf{id}(A) \mid \mathbf{id}[E] \\ A &\rightarrow E \mid E \,;\, A \end{aligned}$$

   (a) Left-factor this grammar so that no two productions with the same left-hand side have right-hand sides with a common prefix.

   (b) Construct an LL(1) parsing table for the left-factored grammar.

   (c) Show the operation of an LL(1) parser on the input string **id(id[id]; id)**.

7. Consider the following CFG, which has the set of terminals $T = \{\mathbf{a}, \mathbf{b}\}$.

$$\begin{aligned} S &\rightarrow X\mathbf{a} \\ X &\rightarrow \mathbf{a} \mid \mathbf{a}X\mathbf{b} \end{aligned}$$

   (a) Construct a DFA for viable prefixes of this grammar using LR(0) items.

   (b) Identify a shift-reduce conflict in this grammar under the SLR(1) rules.

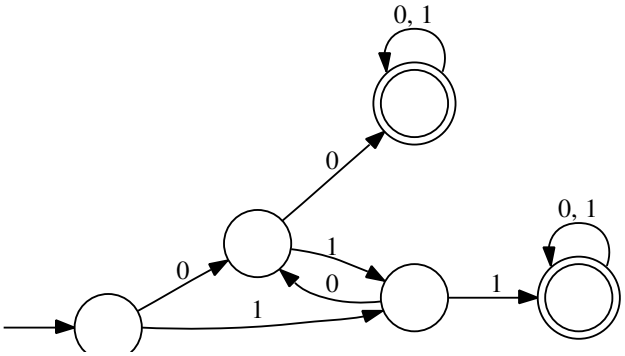
   (c) Assuming that an SLR(1) parser resolves shift-reduce conflicts by choosing to shift, show the operation of such a parser on the input string **aaba**.

   (d) Suppose that the production $X \rightarrow \varepsilon$ is added to this grammar. Identify a reduce-reduce conflict in the resulting grammar under the SLR(1) rules.

Written Assignment 1
Due ??

This assignment asks you to prepare written answers to questions on regular languages, finite automata, and lexical analysis. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

1. Consider the following deterministic finite automaton (DFA) over the alphabet $\Sigma = \{0, 1\}$.



   Give a one-sentence description of the language recognized by the DFA. Write a regular expression for this language.

2. Consider the following languages over the alphabet $\Sigma = \{0, 1\}$.

   - L_1: All strings that contain at least two 0s
   - L_2: All strings that contain at least one 1
   - L_3: All strings that contain at least two 0s and at least one 1
   - L_4: All strings that contain at most one 0 or no 1s

   Give DFAs for each of the languages L_1, L_2, L_3, and L_4.

   Aside: This example illustrates that the regular languages are closed under intersection and complementation. Note that $L_3 = L_1 \cap L_2$ and $L_4 = \Sigma^* - L_3$, where Σ^* represents the language containing all strings over the alphabet Σ.

3. Let E_3 be the language over the alphabet $\Sigma = \{a_1, a_2, a_3\}$ defined as follows.

   E_3: All strings in which a_i occurs an even number of times for some $i \in \{1, 2, 3\}$

   Give a non-deterministic finite automaton (NFA) for the language E_3.

4. Write regular expressions for the following languages over the alphabet $\Sigma = \{0, 1\}$:

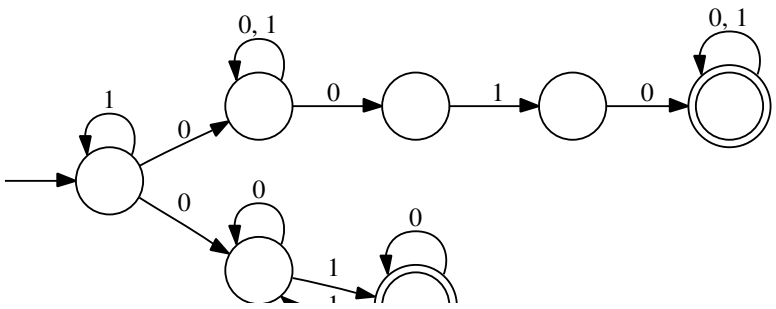
(a) All strings that contain at least one 0 and at least one 1 and that also end with at least two 1s.

(b) All strings that do not begin with 01.

(c) All strings that contain an odd number of 1s.

5. Give a DFA for each of the following languages over the alphabet $\Sigma = \{0, 1\}$.

(a) The language of the following NFA.



(b) The language of the regular expression $(0 + 01)^*1^*$.

6. Consider the string

```
aaabaabbababbb
```

and its tokenization

```
aa a b aabb a b a bbb
```

Give a flex specification with the minimum number of rules that produces this tokenization. Each flex rule should be as simple as possible as well. You may not use regular expression union (i.e., $R_1 + R_2$) in your solution. Do not give any actions; just assume that the rule returns the string that it matches.

Written Assignment 2
Due ??

This assignment asks you to prepare written answers to questions on context-free grammars, parse trees, and parsing. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

1. Give a context-free grammar (CFG) for each of the following languages over the alphabet $\Sigma = \{0, 1\}$:
   (a) All nonempty strings that start and end with the same symbol.
   (b) All strings that contain more 1s than 0s.
   (c) All palindromes (a palindrome is a string that reads the same forwards and backwards).

2. Consider the following CFG.

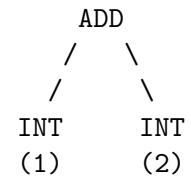
$$\begin{aligned} S &\rightarrow AED \mid F \\ A &\rightarrow Aa \mid a \\ B &\rightarrow Bb \mid b \\ C &\rightarrow Cc \mid c \\ D &\rightarrow Dd \mid d \\ E &\rightarrow bEc \mid bc \\ F &\rightarrow aFd \mid BC \end{aligned}$$

   (a) What is the language generated by this grammar?
   (b) Show that this grammar is ambiguous by giving a string that can be parsed in two different ways. Draw both parse trees.
   (c) Give an unambiguous grammar that generates the same language as the grammar above.

3. The Cool Reference Manual, in Chapter 11, has the context-free grammar defining the syntax for Cool. Create the parse tree from the grammar definition for the following class definition.

```
class FOO inherits BAR {
  i : Int <- 42;
  baz (x:Int) : Foo { i <- x + i };
};
```

   You do not need to construct nonterminals that immediately produce terminals, or include terminals which add no meaning because they are captured by the nature of the production. For example, for the expression “1 + 2”, the parse tree



would be sufficient.

4. Give a one-sentence description of the language generated by each of the following CFGs.

(a)

$$\begin{aligned} S &\rightarrow Z1Z1Z1A \\ Z &\rightarrow Z0 \mid \varepsilon \\ A &\rightarrow A0 \mid A1 \mid \varepsilon \end{aligned}$$

(b)

$$S \rightarrow 0 \mid 1 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1$$

(c)

$$\begin{aligned} S &\rightarrow DC \mid AE \\ A &\rightarrow Aa \mid \varepsilon \\ C &\rightarrow Cc \mid \varepsilon \\ D &\rightarrow aDb \mid \varepsilon \\ E &\rightarrow bEc \mid \varepsilon \end{aligned}$$

5. Consider the following CFG, which has the set of terminals $T = \{\mathbf{id}, (,), [,], ;\}$.

$$\begin{aligned} E &\rightarrow \mathbf{id} \mid \mathbf{id}(A) \mid \mathbf{id}[E] \\ A &\rightarrow E \mid E \,;\, A \end{aligned}$$

(a) Left-factor this grammar so that no two productions with the same left-hand side have right-hand sides with a common prefix.
(b) Construct an LL(1) parsing table for the left-factored grammar.
(c) Show the operation of an LL(1) parser on the input string **id(id[id]; id)**.

6. Consider the following CFG, which has the set of terminals $T = \{\mathbf{a}, \mathbf{b}\}$.

$$\begin{aligned} S &\rightarrow X\mathbf{a} \\ X &\rightarrow \mathbf{a} \mid \mathbf{a}X\mathbf{b} \end{aligned}$$

(a) Construct a DFA for viable prefixes of this grammar using LR(0) items.
(b) Identify a shift-reduce conflict in this grammar under the SLR(1) rules.
(c) Assuming that an SLR(1) parser resolves shift-reduce conflicts by choosing to shift, show the operation of such a parser on the input string **aaba**.
(d) Suppose that the production $X \rightarrow \varepsilon$ is added to this grammar. Identify a reduce-reduce conflict in the resulting grammar under the SLR(1) rules.