Assignment 7

COMP 2230
Due: December 2, 2026 by 11:59 p.m.
Upload your Word, PDF, or LaTeX file(s) to Brightspace

Exercises from the Textbook (50 points):

Section 12.1: 5, 26, 27, 29, 41

Section 12.2: 13, 27, 48

Section 12.3: 2, 8

Additional Exercises (50 points):

1. You're the CTO of a start-up company that uses Java to predict stock valuations. One morning, your CEO is excited after talking to a new software vendor selling a tool that can analyze Java code and perfectly determine whether it will throw any exceptions for a range of possible inputs. A tool like that could make the code that your teams produce much more reliable. What is your response to your CEO?

2. Write a regular expression that accepts IPv4 addresses, which contain four integer values between the values of 0 and 255 separated by three periods. There are ranges of addresses that are reserved for particular purposes, but you don't need to worry about those.

   Examples of legal IPv4 addresses:

   ```
   192.168.1.1
   0.0.0.0
   255.255.255.255
   ```

   Examples of illegal IPv4 addresses:

   ```
   192.168.1.
   127..127.127
   256.100.100.100
   192.168.1.1.1
   ```

3. A **`double`** literal in ISO C90 is a sequence of decimal digits that also contains a decimal point, an **`E`** (or **`e`**) to mark scientific notation, or both. It can optionally start with a minus to mark it as negative. If it is in scientific notation, it must have at least one digit before and after the **`E`** (or **`e`**). The digits after the **`E`** (or **`e`**) may optionally begin with a minus to mark the exponent as negative. A legal literal may only have a single decimal point, and the decimal point must not occur after the **`E`** (or **`e`**). A legal literal must always contain at least one decimal digit.

   Examples of legal **`double`** literals:

   ```
   1.
   .1
   0.0
   -5E8
   .5E8
   5.E8
   -13.8732e-207
   ```

   Examples of illegal **`double`** literals:

   ```
   1
   .
   2.1.3
   5-E8
   6E4E2
   6E2.3
   ```

   Write a regular expression for legal ISO C90 **`double`** literals.

4. Now, draw an equivalent finite-state automaton for ISO C90 double literals.

5. Regular expressions and finite automata are useful tools, but there are many simple languages that cannot be expressed by them. To express more complicated languages, one possibility is to use a **context free grammar**. A context free grammar expresses the possible strings in a given alphabet based on a finite number of rules relating non-terminal symbols to terminal symbols. A terminal is simply a symbol from alphabet Σ. A non-terminal is simply a placeholder symbol that can be expanded to some other sequence of symbols.

   Each rule in a context free grammar maps a non-terminal symbol to any sequence of terminal and non-terminal symbols. A single non-terminal may have many rules, any of which could be chosen as needed. Context free grammars typically start with a single special non-terminal S.

   For example, here is a context free grammar for the language that is any positive number of repetitions of the string ab (assuming $\Sigma = \{a, b\}$):

   $S \rightarrow B$
   $B \rightarrow abB$
   $B \rightarrow ab$

   This context free grammar is equivalent to the regular expression $ab(ab)^*$ or simply $(ab)^+$. Although all regular languages can be expressed with a context free grammar, not all context free languages can be expressed as a regular language. Assuming $\Sigma = \{a, b\}$, write a context free grammar for the language in which all strings contain the same number of a's as b's, a language which is **not** regular.