CS 427: Problem Set #1

Due on September 18, 2017 at 5:50pm

Professor Zerksis D. Umrigar

Tim Hung

Problem 1

Discuss how you would extend the implementation of tl0 discussed in class to support a unary minus operator to allow evalution of expressions like '1 - -2'. It is not necessary to show code but sufficient to precisely describe how you would change the implementation. "15-points".

Solution

Problem 2

I once used a proprietary programming language where the programming language manual had a strange restriction. It went something like:

"Integers can have a value between -2147483648 through 2147483647 inclusive. However, the literal value -2147483648 cannot be present in the program; all other integer literal values can be present in the program."

Discuss possible reasons for this strange restriction.

Hint: Our implementation of tl0 is subject to similar restrictions but not our implementation of tl1 (the difference is *not* due to our use of Ruby for implementing tl1). "15-points"

Solution

Problem 3

Write regular expressions for each of the following:
Your answers should use the regex syntax discussed in class. "15-points"

(a) Strings over $\{a, b, c\}$ with an even number of a's.

(b) Strings delimited using double quote characters ", where a string can contain any character except double quotes and can even contain double quotes if such contained double quotes are doubled. Examples of legal strings include the string "" of length 0, the string "a" of length 1, and the string """" of length 1 containing a single double-quote character.

(c) Positive binary numbers n without leading zeros such that there exist positive integers a, b and c with $a^n + b^n = c^n$.

Problem 4

In a string of length n, how many of the following are there?

(a) prefixes

Every substring that starts at index 0 and ends at index $i \in \{1, n\}$ is a prefix.

$$n$$

(b) substrings

For each substring, we choose a start index and an end index.

There are $\binom{n}{2}$ ways to choose two indices from n characters.

$$\binom{n}{2}$$

(c) subsequences

A subsequence is comprised of any element of the powerset of the string.

The size of the powerset of a set of size n is

$$2^n$$

Problem 5

Why would it be a problem if a programming language had 2 different binary operators $\oplus$ and $\otimes$ having the same precedence, but $\oplus$ was left associative while $\otimes$ was right associative? "0-points"

Solution

Problem 6

Write "regular definitions" to describe the following languages:
Your answers should use the regex syntax discussed in class. "15-points"

Solution

(a) All strings of balanced parentheses with a maximum nesting depth of 2. Examples include ϵ, (), (())() and (()())().

(b) All strings of balanced parentheses. Example include ϵ, (), (())() and ((((()))))().

(c) C floating point constants as defined on pg 194 of the "The C Programming Language" book:

A floating constant consists of an integer part, a decimal point, a fraction part, an 'e' or 'E', an optionally signed integer exponent and an optional type suffix, one of 'f', 'F', 'l' or 'L'. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the 'e' and the exponent (not both) may be missing.

Problem 7

Consider the Java java.util.regex package:

Solution

(a) It provides 3 different kinds of quantifiers: "greedy quantifiers", "reluctant quantifiers" and "possesive quantifiers". Give situations where each of these different types of quantifiers may be useful.

Problem 7 continued on next page...

(b) How would you use this package to write a scanner for a programming language. Consider as an example, a language which contains tokens for numbers, identifiers and strings, with whitespace.and comments ignored (the exact details of the lexical syntax being unimportant). Your proposal should specifically address the following:

- How would you set things up so that for each recognized lexeme you would only make a single call to the Java RE engine?
- After the call to the Java RE engine completes, how would you identify which kind of token has been matched?
- Which of the several calls in the 'Matcher' class would you use to do the matching?
- How would you structure a 'nextToken()' function?

"15-points"