CMSC 22100, Spring 2020: Quiz

Please complete this quiz either by writing by hand and pushing a photograph or scan of your responses to your repository, or using a computer strictly as a typewriter and doing the same.

By submitting the quiz you are implicitly certifying that you have scrupulously adhered to the standards of academic honesty. In particular, you are confirming that

- you did not study in advance of taking the quiz any time between noon CDT on Wednesday May 6 and noon CDT on Thursday May 7,

- you wrote your responses in one 40-minute sitting (or more time if by prior special arrangement) in the absence of any study materials ("closed book"), and,

- you did not use a computer to code-edit and/or compile any programs to check their correctness in any way.

No one on the staff will answer quiz questions during the 24-hour window, and you must of course refrain from communicating with one another about the quiz during that time period.

Your responses must be added, committed, and pushed by Thursday, May 7, at noon CDT. Save your response file(s) (do try to keep it to one file if possible) into the directory quiz/ in your 221 repository. The file doesn't need a particular name, although something like "quiz.pdf" or "quiz.txt" seems natural. Do be sure to put it in the quiz/ directory.

==

1) Write a Standard ML datatype for abstract syntax trees corresponding to the following term grammar.

```
t ::= true
    | false
    | maybe
    | (! t)
    | (+ t t)
    | (? t t t)
```

2) The exclusive-or operator is a binary operator on booleans such that the result is true when exactly one of the operands is true, and is false otherwise.

   The mini-language X consists only of boolean constants and xor (exclusive or) expressions. Here is its term grammar and value grammar.

```
t ::= #t | #f | (xor t t)
v ::= #t | #f
```

   Write small-step evaluation rules for X so that xor expressions, when evaluated all the way to normal forms, yield the usual exclusive-or results.

3) The following parenthesis-counting algorithm can determine whether the parentheses in a given string of characters are balanced or not. It is not quite as simple as counting the number of each

parenthesis and making sure the counts are equal.

Start with a counter at 0. Traverse each character in the string left to right. Whenever you encounter a left parenthesis, increment the counter, and whenever you encounter a right parenthesis, decrement the counter. The counter is unaffected by any other character. When you reach the end of the string, the counter should be 0. If it is not, then the string contains unbalanced parentheses. Furthermore, at no point in the counting should the counter become negative. If that ever happens, you have a too-early right parenthesis in the string, and the parentheses are unbalanced.

Here are some examples to try as thought experiments.

```
"(x)"
"(x) (x)"
"((y))"
"((x)"
"(y))"
")("
"() )( ()"
```

Write an implementation of this algorithm in Standard ML. Name the function "balanced" with the following type:

```
balanced : string -> bool
```

The function only needs to check parentheses, not other delimiters, and should never raise an exception. Writing one or more helper functions is recommended.