

Name: _____

COSC 8, Fall 2007

Practice Final Exam

December 7, 2007

Problem	1	2	3	4	5	Total
Grade						
Points	20	25	20	15	20	100
Grader						

General Instructions

There are five (5) problems on this exam. Please check now that you have a complete exam booklet with eight (8) numbered pages. Please write your name on *each page* of the exam.

Be sure to look at *all* of the problems. Some are more difficult than others. Don't waste a lot of time on a problem that is giving you a hard time—if you get stuck, move on to another problem, and then return to it later.

There is space provided to answer each question, and you may request extra paper if you so desire. If you do use extra paper to record your solutions, be sure to write your name prominently on each extra page, and clearly indicate which problem is being answered there. You do not need to turn in any extra scratch-paper you use while working out your solutions; only the pages with your answers on them need to be submitted.

To help insure that you don't accidentally miss any of the questions, each section where an answer is requested has been marked with a \Leftarrow symbol in the right margin.

Please write neatly—the course staff reserves the right to ignore illegible answers. When writing solutions that involve program code, proper indentation of your code is important!

This examination covers material through Lecture 28.

<p>You may not consult any books, notes, study-guides, or other outside material during the exam period, except for one sheet of 8.5 by 11 inch notebook paper which you are allowed to make notes on (both sides). The notes may be hand-written or typed in 10 point or bigger font. You will also be supplied with a mini-manual of list functions.</p>
--

1. (20 Points) **Streams.**

Consider a stream of the rows of Pascal's triangle, with the rows given as lists:

`[[1], [1,1], [1,2,1], [1,3,3,1], [1,4,6,4,1], ...]`

To get each row, you add two copies of the previous row, but one copy is shifted by one place.

Consider the entry `[1,3,3,1]`:

$$\begin{array}{r} 1 \quad 3 \quad 3 \quad 1 \\ + \quad 1 \quad 3 \quad 3 \quad 1 \\ \hline 1 \quad 4 \quad 6 \quad 4 \quad 1 \end{array}$$

Define a stream `pascalTri` that uses this approach to recursively generate itself.

2. (25 Points) Parsing.

Regular expressions are used in computer science to specify the format of things like variables, numbers, etc. A simple form of regular expression can be defined by the following BNF:

```
regExpr = letter
         | '(' regExpr regExpr ')'
         | '(' regExpr '|' regExpr ')'
         | '(' regExpr ')' '*'
```

A regular expression tree has a different constructor for each of the options in the BNF above:

```
data RegExpr = Literal Char
              | Seq RegExpr RegExpr
              | Or RegExpr RegExpr
              | Star RegExpr
  deriving Show
```

Write a parser `regExpr :: Parser RegExpr` that parses an input string into a regular expression tree. You may assume that `ParserX` has been imported, and may use any of the things that it exports:

```
module ParserX(Parser,
               (#), (-#), (#-), (#:), (#+), (!), (>->), (?), lit, letter, space,
               char, takeParse, takeWhileParse, cons, returnParse, failParse,
               takeRepeatParse, parseSpaces, parseWord, wordChoice, elimSpacesAround,
               module Data.Char)
```

Thus a call to `regExpr "((a)*((bc)|d))*"` should return

```
Just (Star (Seq (Star (Literal 'a'))
                (Or (Seq (Literal 'b') (Literal 'c')) (Literal 'd'))), "")
```

Write your parser here:

3. (20 Points) **Databases.**

(a) Describe how to take the inner join of two tables. What property does the column chosen in the first table have to have? If the first table has m rows and the second table has n rows, what is the maximum number of rows that their inner join can have? \Leftarrow

(b) Describe how to take the join of two tables. If the first table has m rows and the second table has n rows, what is the maximum number of rows that their join can have? \Leftarrow

4. (15 Points) **Regions and Pictures.**

(a) (10 Points) User Coordinates

Consider the following functions for converting user coordinates to window coordinates:

```
userXtoWin    :: UserWindow -> Float -> Int
userXtoWin uw x = round (intToFloat (xWin uw) * (x - userXmin uw)/userXrange uw)
```

```
userYtoWin    :: UserWindow -> Float -> Int
userYtoWin uw y = round (intToFloat (yWin uw) * (userYmax uw - y)/userYrange uw)
```

Here `xWin uw` and `yWin uw` are the width and height of the graphics window expressed in pixels, `userXmin uw` and `userYmax uw` are the minimum x value and maximum y value in the user window, and `userXrange uw` and `userYrange uw` are the width and height of the window expressed in user coordinates.

Explain why these functions correctly convert x and y values in user coordinates to the corresponding pixel values in window coordinates. In particular explain why one formula uses $(x - \text{userXmin } uw)$ and the other uses $(\text{userYmax } uw - y)$. Drawing a picture may help.

(b) (5 Points) Tetris

In the sample solution to SA9 I kept track of the (x, y) coordinates of the center of the piece as it fell and was moved left and right. I translated the piece to the appropriate position before I drew it every time through the loop. Both `x`, `stime`, and the original piece are passed to recursive calls to `loop`. Moving the piece left or right involved updating `x`, not translating the piece.

On the other hand, when rotating the piece I simply passed (`RotateL piece`) to the recursive call to `loop` rather than keeping a count of rotations and passing that. I could have used the same strategy for changing (x, y) values. Instead of passing `x`, `stime`, and the original piece I could have passed the current time and a translated piece.

Why might this alternate design be worse than the one I used?



5. (20 Points) **Short Answer.**

- (a) What is pruning in backtracking? Where does the name come from? Why does it speed up a backtracking program? ⇐

- (b) The List Monad is useful in non-determinism. In the second problem set you wrote a function `stepNFA` that took a set of states and generated all successor states after one step of the NFA. One way to write the code is:

```
stepNFA :: (Eq a, Eq b) => Automaton a b -> [a] -> b -> [a]
stepNFA nfa states label =
    uniques $ foldr (++) [] (map (transNFA nfa label) states)
```

where `uniques` eliminates duplicates in a list and `(transNFA nfa label)` is a function that takes a state and returns a list of states that are reachable in one step from that state following edges labeled `label`. Its principal type is `a -> [a]`, where `a` is the type of a vertex in `nfa`.

Show how to use the List Monad given below to replace everything after the “\$” in the definition of `stepNFA` by a simple monad operation. Unfold the monad command to show that it does the correct thing.

```
instance Monad [] where
m >>= k      = concat (map k m)
return x     = [x]
fail x       = [ ]
```

⇐

(c) Describe the criteria used in Norvig's spell checker to pick the word most likely to be the correct spelling of a mis-spelled word.



(d) What would happen if the queue were replaced by a stack in BFS? Would everything reachable be found? Would the tree returned be a shortest path tree?

