Name: ____________________

CSCI 291 — Spring 2023

Midterm Exam

This exam should have four pages. Feel free to write helper functions as part of any of your solutions. Comments and type signatures are not required for full credit.
Closed book and notes. No calculators allowed or necessary.

Problem 1: [10 points]

Haskell's pure mathematical functions aren't allowed to produce any side effects. List at least two important benefits achieved by disallowing side effects.

Problem 2: [10 points]

Given what you know about lazy evaluation, will the following expression evaluate successfully? (That is, will it produce a value in a finite amount of time without throwing an exception or otherwise crashing?) If not, why can't lazy evaluation save us? If so, what does it produce and how does lazy evaluation help?

```
head (map (10 `div`) [10,9..])
```

Problem 3: [15 points] Define the function `shift`. It's similar to the scale function we wrote in class, but instead of scaling a list of numbers so it fits in the range between 0 and 100, it should simply shift each input number by the same amount such that the largest number becomes 100. Feel free to use explicit recursion or not, as you prefer. You may assume the input list contains at least one value.

```
> shift [10,30,20]
[80,100,90]
> shift [80,100,120]
[60,80,100]
> shift [-20, 90]
[-10,100]
```

Problem 4a: [15 points] The Evil Hangman assignment required that you generate a pattern of dashes and letters given a word and the letters guessed so far. Below, define the `pattern` function which takes a more general approach: It takes a Boolean-valued *function* and a word, and replaces any characters that "pass the test" with dashes. For full credit, your solution should be *recursive*.

```
> pattern (=='a') "aardvark"
"--rdv-rk"
> pattern (< 'e') "aardvark"
"--r-v-rk"
> pattern ((=='e').succ) "aardvark"
"aar-vark"
```

Problem 4b: [15 points] Solve the same problem again, but this time define `pattern` *without* using explicit recursion. (That is, use only built-in tools like map, filter, fold, list comprehensions, etc.)

Problem 4c: [10 points] What type would Haskell infer for the function above? That is, what would it report if you entered `:type` pattern in ghci? Try to use the same notation that Haskell would. (Note: You can answer this question even if you weren't able to define the function in the previous parts.)

Problem 5: [25 points]

The questions below apply to this mysterious function. (Recall that `elem` is the built-in function that tests whether an item appears in a list, and `fst` and `snd` retrieve items from tuples.)

```
mystery ps = foldr1 (||) group3
  where
    group1 = map fst ps
    group2 = map snd ps
    group3 = map (`elem` group1) group2
```

5a) If `mystery` is passed `[("ok","good"),("bad","fine"),("good","great")]`, what would be stored in the variable `group1`?

5b) If `mystery` is passed `[("ok","good"),("bad","fine"),("good","great")]`, what would be stored in the variable `group2`?

5c) What would `mystery` return for the input from 5a and 5b?

5d) Describe, in *English*, what the `mystery` function does in general.