Homework 2 – Fall 2022 (due: October 13, 2022)

This homework is meant to prepare you for upcoming exams. You will be expected to be familiar with the concepts covered in this homework and be able to write grammatically-correct Haskell code. Each question has a time estimate; you'll know you're ready for the exam when you can solve them roughly within their time constraints.

You must turn in a PDF file with your answers via Gradescope - you may include both typed and hand-written solutions, so long as they are legible and make sense to our TAs. Make sure to clearly label each answer with the problem number you're solving so our TAs can more easily evaluate your work.

For all questions where you are writing Haskell code, you should give your functions a type definition.

1.

a) (2 min.) Use the `map` function to write a Haskell function named `scale_nums` that takes in a list of `Integers` and an `Integer` named `factor`. It should return a new list where every number in the input list has been multiplied by `factor`.

Example:
`scale_nums [1, 4, 9, 10] 3` should return `[3, 12, 27, 30]`.

You may not define any nested functions. Your solution should be a single, one-line map expression that includes a lambda.

b) (2 min.) Use the `filter` and `all` functions to write a Haskell function named `only_odds` that takes in a list of `Integer` lists, and returns all lists in the input list that only contain odd numbers (in the same order as they appear in the input list). Note that the empty list vacuously satisfies this requirement.

Example:

`only_odds [[1, 2, 3], [3, 5], [], [8, 10], [11]]` should return `[[3, 5], [], [11]]`.

You may not define any nested functions. Your solution should be a single, one-line `filter` expression that includes a lambda.

c) (2 min.) In Homework 1, you wrote a `largest` function that returns the larger of two words, or the first if they are the same length:

```
largest :: String -> String -> String
largest first second =
    if length first >= length second then first else second
```

Use one of `foldl` or `foldr` and the `largest` function to write a Haskell function named `largest_in_list` that takes in a list of `String`s and returns the longest `String` in the list. If the list is empty, return the empty string. If there are multiple strings with the same maximum length, return the one that appears first in the list. **Do not use the `map`, `filter` or `maximum` functions in your answer.**

Your answer should be a single, one-line `fold` expression.

Example:
`largest_in_list ["how", "now", "brown", "cow"]` should return `"brown"`.
`largest_in_list ["cat", "mat", "bat"]` should return `"cat"`.

2.

a) (5 min.) Write a Haskell function named `count_if` that takes in a predicate function of type `(a -> Bool)` and a list of type `[a]`. It should return an `Int` representing the number of elements in the list that satisfy the predicate. **Your solution must use recursion. Do not use the `map`, `filter`, `foldl`, or `foldr` functions in your solution.**

Examples:

`count_if (\x -> mod x 2 == 0) [2, 4, 6, 8, 9]` should return `4`.

`count_if (\x -> length x > 2) [“a”, “ab”, “abc”]` should return `1`.

b) (3 min.) Now, reimplement the same function above (call it `count_if_with_filter`), **but use the `filter` function in your solution**.

c) (3 min.) Now, reimplement the same function above (call it `count_if_with_fold`) **but use either `foldl` or `foldr` in your solution**.

3.

a) (3 min.) Explain the difference between currying and partial application.

b) (4 min.) Suppose we have a Haskell function with type `a -> b -> c`. Consider the other two function types:

i. `(a -> b) -> c`

ii. `a -> (b -> c)`

Is `a -> b -> c` equivalent to i, ii, both, or neither? Why?

c) (2 min.) Consider the following Haskell function:

```
foo :: Integer -> Integer -> Integer -> (Integer -> a) -> [a]
foo x y z t = map t [x,x+z..y]
```

Rewrite the implementation of `foo` as a chain of lambda expressions that each take in **one** variable to demonstrate the form of a curried function.

4. Consider the following Haskell function:

```
f a b =
  let c = \a -> a   -- (1)
      d = \c -> b   -- (2)
  in \e f -> c d e  -- (3)
```

a) (1 min.) What variables (if any) are captured in the lambda labeled (1)?

b) (1 min.) What variables (if any) are captured in the lambda labeled (2)?

c) (1 min.) What variables (if any) are captured in the lambda labeled (3)?

d) (4 min.) Suppose we invoke f in the following way:

```
f 4 5 6 7
```

What are the names of the variables that the passed in values (4, 5, 6, and 7) are bound to? Which of them (if any) are actually referenced in the implementation of f? Explain.

5. (5 min.) C allows you to point to functions, as in the following code snippet:

```c
int add(int a, int b) {
  return a + b;
}

int main() {
  // Declare a pointer named `fptr` that points to a function
  // that takes in two int arguments and returns another int
  int (*fptr)(int, int);

  // Assign the address of the `add` function to `fptr`
  fptr = &add;

  // Invoke the function using `fptr`. This returns 8.
  (*fptr)(3, 5);
}
```

Function pointers are first-class citizens like any other pointer in C: they can be passed as arguments, used as return values, and assigned to variables. As a reminder, however, C does not support nested functions.

Compare function pointers in C with closures in Haskell. Are Haskell closures also first-class citizens? What (if any) capabilities do function pointers have that closures do not (and vice versa)?

6. Haskell allows you to define your own custom data types. In this question, you'll look at code examples and use them to write your own (that is distinct from the ones shown).

   Consider the following code example:

```haskell
data Triforce = Power | Courage | Wisdom

wielder :: Triforce -> String
wielder Power = "Ganon"
wielder Courage = "Link"
wielder Wisdom = "Zelda"

princess = wielder Wisdom
```

a) (2 min.) Define a new Haskell type `InstagramUser` that has two value constructors (without parameters) - `Influencer` and `Normie`.

b) (2 min.) Write a function named `lit_collab` that takes in two `InstagramUsers` and returns `True` if they are both `Influencers` and `False` otherwise.

Consider the following code example:

```
data Character = Hylian Int | Goron | Rito Double | Gerudo |
Zora

describe :: Character -> String
describe (Hylian age) = "A Hylian of age " ++ show age
describe Goron = "A Goron miner"
describe (Rito wingspan) = "A Rito with a wingspan of " ++ show
wingspan ++ "m"
describe Gerudo = "A mighty Gerudo warrior"
describe Zora = "A Zora fisher"
```

c) (2 min.) Modify your `InstagramUser` type so that the `Influencer` value constructor takes in a list of Strings representing their sponsorships.

d) (3 min.) Write a function `is_sponsor` that takes in an `InstagramUser` and a `String` representing a sponsor, then returns `True` if the user is sponsored by `sponsor` (this function always returns `False` for `Normies`).

Consider the following code example:

```
data Quest = Subquest Quest | FinalBoss

count_subquests :: Quest -> Integer
count_subquests FinalBoss = 0
count_subquests (Subquest quest) = 1 + count_subquests quest
```

e) (2 min.) Modify your `InstagramUser` type so that the `Influencer` value constructor also takes in a list of other `InstagramUsers` representing their followers (after their sponsors).

f) (3 min.) Write a function `count_influencers` that takes in an `InstagramUser` and returns an `Integer` representing the number of `Influencers` that are following that user (this function always returns `0` for `Normies`).

g) (2 min.) Use GHCi to determine the type of `Influencer` using the command `:t Influencer`. What can you infer about the type of custom value constructors?

7. Consider the following Haskell data type:

```
data LinkedList = EmptyList | ListNode Integer LinkedList
  deriving Show
```

a) (3 min.) Write a function named `ll_contains` that takes in a `LinkedList` and an `Integer` and returns a `Bool` indicating whether or not the list contains that value.

Examples:

`ll_contains (ListNode 3 (ListNode 6 EmptyList)) 3`

should return `True`.

`ll_contains (ListNode 3 (ListNode 6 EmptyList)) 4`

should return `False`.

b) (3 min.) We want to write a function named `ll_insert` that inserts a value at a given zero-based index into an existing `LinkedList`. Provide a type definition for this function, explaining what each parameter is and justifying the return type you chose.

c) (5 min.) Implement the `ll_insert` function. If the insertion index is 0 or negative, insert the value at the beginning. If it exceeds the length of the list, insert the value at the end. Otherwise, the value should have the passed-in insertion index after the function is invoked.

8. In this question, we'll examine how the choice of programming language/paradigm can affect the difficulty of a given task.

a) (5 min.) Using C++, write a function named `longestRun` that takes in a `vector` of booleans and returns the length of the longest consecutive sequence of `true` values in that vector.

Examples:
Given `{true, true, false, true, true, true, false}`, `longestRun(vec)` should return `3`.
Given `{true, false, true, true}`, `longestRun(vec)` should return `2`.

b) (10 min.) Using Haskell, write a function named `longest_run` that takes in a list of `Bool`s and returns the length of the longest consecutive sequence of `True` values in that list.

Examples:
`longest_run [True, True, False, True, True, True, False]` should return `3`.
`longest_run [True, False, True, True]` should return `2`.

c) (10 min.) Consider the following C++ class:

```
#import <vector>
using namespace std;

class Tree {
public:
  unsigned value;
  vector<Tree *> children;

  Tree(unsigned value, vector<Tree *> children) {
    this->value = value;
    this->children = children;
  }
};
```

Using C++, write a function named `maxTreeValue` that takes in a `Tree` pointer `root` and returns the largest value within the tree. If `root` is `nullptr`, return `0`.

This function may not contain any recursive calls.

d) (5 min.) Consider the following Haskell data type:

```
data Tree = Empty | Node Integer [Tree]
```

Using Haskell, write a function named `max_tree_value` that takes in a `Tree` and returns the largest `Integer` in the `Tree`. Assume that all values in the tree are non-negative. If the root is `Empty`, return `0`.

Example:
`max_tree_value (Node 3 [(Node 2 [Node 7 []]), (Node 5 [Node 4 []])])` should return 7.

9. (10 min.) Write a Haskell function named `fibonacci` that takes in an `Int` argument `n`. It should return the first `n` numbers of the Fibonacci sequence (for this problem, we'll say that the first two numbers of the sequence are 1, 1).

   Examples:

   `fibonacci 10` should return `[1,1,2,3,5,8,13,21,34,55]`.

   `fibonacci -1` should return `[]`.

   Hint: You may find it easier to build the list in reverse in a right-to-left manner, then use the reverse function.

10. (20 min.) Super Giuseppe is a hero trying to save Princess Watermelon from the clutches of the evil villain Oogway. He has a long journey ahead of him, which is comprised of many events:

```
data Event = Travel Integer | Fight Integer | Heal Integer
```

Super Giuseppe begins his adventure with 100 hit points, and may never exceed that amount. When he encounters:
A `Travel` event, the `Integer` represents the distance he needs to travel. During this time, he heals for ¼ of the distance traveled (floor division).
A `Fight` event, the `Integer` represents the amount of hit points he loses from the fight.
A `Heal` event, the `Integer` represents the amount of hit points he heals after consuming his favorite power-up, the bittermelon.

If Super Giuseppe has 40 or fewer life points after an Event, he enters defensive mode. While in this mode, he takes half the damage from fights (floor division), but he no longer heals while traveling. Nothing changes with Heal events. Once he heals **above** the 40 point threshold following an Event, he returns to his normal mode (as described above).

Write a Haskell function named `super_giuseppe` that takes in a list of `Events` that comprise Super Giuseppe's journey, and returns the number of hit points that he has at the end of it. If his hit points hit 0 or below at any point, then he has unfortunately Game Over-ed and the function should return `-1`.

Examples:
`super_giuseppe [Heal 20, Fight 20, Travel 40, Fight 60, Travel 80, Heal 30, Fight 40, Fight 20]` should return `10`.
`super_giuseppe [Heal 40, Fight 70, Travel 100, Fight 60, Heal 40]` should return `-1`.