Assignment No. 7
EECS 468
Programming Language Paradigms
Due: 11:59 PM, Monday, November 13, 2023
Submit deliverables in a single zip file to Canvas
Files in other formats (e.g., .tar) will not be graded
Name of the zip file: FirstnameLastname_Assignment7 (with your first and last name)
Name of the Assignment folder within the zip file: FirstnameLastname_Assignment7

Deliverables:

1. Copy of Rubric7.docx with your name and ID filled out (do not submit a PDF)
2. Haskell script file(s).
3. Screen print(s) as described below (Copy and paste the output to a Word document and PDF it).

Assignment:

- In this assignment, you will define 5 Haskell functions.
- Each function description below shows an example of the function working.
- Once your code is working, show that the example works on your function with a screen print.
- Also, show in a screen print how your function executes using a test case.
- You may put all of the functions in one script file or in separate script files.
- You may put the screen prints in one file or separate files.
- **replicate'** function:

  In a similar manner to the function **length** described in the "Haskell List Comprehension" lecture, show how the library function: **replicate :: Int -> a -> [a]**, which produces a list of identical elements, can be defined using list comprehension. (Name the new function **replicate'**, so Haskell will not give you an error). For example:

  > replicate' 3 True
  [True, True, True]

  Show a screen print with your code using the following test case:

  > replicate' 5 "test code"

- **perfects** function:

  A positive integer is perfect if it equals the sum of all of its factors, excluding the number itself. Using a list comprehension, define a function: **perfects :: Int -> [Int]** that returns the list of all perfect numbers up to a given limit. For example:

  > perfects 500
  [6,28,496]

  Show a screen print with your code of:

  > perfects 9000

- **find** function:

  Suppose that we represent a lookup table by a list of pairs of keys and values. Then for any type of keys that supports equality, define a function

as follows: **find :: Eq a => a -> [(a,b)] -> [b]** that returns the list of all values that are associated with a given key in a table. For example:

> find 'b' [('a',1),('b',2),('c',3),('b',4)]
[2,4]

Show a screen print with your code of:

> find 'c' [('a',1),('b',2),('c',3),('b',4),('c',25)]

- **positions** function:

  Redefine the **positions** function from the "Haskell List Comprehension" lecture using the **find** function. For example:

  > positions 0 [1,0,0,1,0,1,1,0]
  [1,2,4,7]

  Show a screen print with your code using the following test case:

  > positions 1 [1,0,0,1,0,1,1,0]

- **scalarproduct** function:

  The scalar product of two lists of integers xs and ys of length n is given by the sum of the products of the corresponding integers:

  $$\sum_{i=0}^{n-1} (xs_i * ys_i)$$

  For example:

  > scalarproduct [1,2,3] [4,5,6]
  32

  Using a list comprehension and the zip function, define a function that returns the scalar product of two lists.

  Show a screen print with your code using the following test case:

  > scalarproduct [-1,2,3] [-4,-5,6]

- Provide comments for the Haskell code that explain what each line of code is doing. See rubric below.

| **Rubric for Program Comments** | | |
|---|---|---|
| **Exceeds Expectations (90-100%)** | **Meets Expectations (80-89%)** | **Unsatisfactory (0-79%)** |
| Software is adequately commented with prologue comments, comments summarizing major blocks of code, and comments on every line. | Prologue comments are present but missing some items or some major blocks of code are not commented or there are inadequate comments on each line. | Prologue comments are missing all together or there are no comments on major blocks of code or there are very few comments on each line. |

Adequate Prologue Comments:

- Name of program contained in the file (e.g., EECS 468 Assignment 7 - Replicate)
- Brief description of the program, e.g.:
  - Haskell function for replicate
- Inputs,e.g.,:

    - Number of replications
    - Element to replicate
- Output, e.g.,
    - List of replicated elements
- All collaborators
- Other sources for the code ChatGPT, stackOverflow, etc.
- Author's full name
- Creation date: The date you first create the file, i.e., the date you write this comment

Adequate comments summarizing major blocks of code and comments on every line:

- Provide comments that explain what each line of code is doing.
- You may comment each line of code (e.g., using --) and/or provide a multi-line comment (e.g., using {- and -}) that explains what a group of lines does.
- Multi-line comments should be detailed enough that it is clear what each line of code is doing.
- Each block of code must indicate whether you authored the code, you obtained it from one of the sources listed in the prolog, or one of your collaborators authored the code, or if it was a combination of all of these.

Collaboration and other sources for code:

- When you collaborate with other students or use other sources for the code (e.g., ChatGPT, stackOverflow):
    - Your comments must be significantly different from your collaborators.
    - More scrutiny will be applied to grading your comments in particular explaining the code "in your own words", not the source's comments (e.g., ChatGPT's comments).
- Failure to identify collaborators or other sources of code will not only result in a 0 on the assignment but will be considered an act of Academic Misconduct.
- Students who violate conduct policies will be subject to severe penalties, up through and including dismissal from the School of Engineering.