Homework 02

Answer the questions below, and submit electronically via elearning. Make sure you submit a couple hours early at the latest and double check your submission to ensure everything is in order *before* the submission time. Your answers should be submitted as a ".doc", ".docx", or ".pdf" file. Your answers should be typed, but scans or pictures of hand drawn figures and diagrams may be included in the file when needed. Code should be turned in as plain text source code. Haskell files use the ".hs" extension. All funtions yield a value. No printing is needed for any of the questions unless explicitly stated. Check out the standard library documentation. Link in the "Class Notes" section on e-learing.

Due: Friday, February 23rd 11:59pm

Chapter 15: Functional Programming Languages

1. **(5pt)** Explain how recurssion can be ineffecient, and how tail-call optimization solves it.

2. **(10pt)** The following code is not tail recurssive.

   ```
   range x y
     | x > y = []
     | x == y = [y]
     | otherwise = x :  range (x+1) y
   ```

   Analyse this code. What is its type? What does it do? Does it consider every case? Explain why such code is not ineffecient when written in Haskell.

3. **(10pt)** Consider a list where every element is a pair of Ints, where the first element of eachpair is either a 0 or a 1. An example input is written below.

   ```
   [(0,1),(1,2),(1,3),(0,4),(0,3)]
   ```

   For the purposes of this question, let's call the first element of each pair the *key* and the second element of each pair the *value*. Now consider a function, `sumByKey`, that transforms such a list to a pair where

   - the first element is the sum of the values of all pairs with 0 as the key, and
   - the second element is the sum of the values of all pairs with 1 as the key.

   Since the key can only be zero or one, any other value is an error. Make sure to handle errors properly. The function should be tail-recursive. It may be helpful to create helper functions. Also do not forget about map and filter.

4. **(5pt)** Consider the function `sumPairs` that takes a list of Ints and transforms them into a new list of Ints by adding the numbers pair-wise. That is the first element of

the resulting list is the sum of the first two elements of the input, the second element of the resulting list is the sum of the 3rd and 4th elements of the input, and so on. If there is an odd number of elements, then the last element remains unchanged. As an example, `sumPairs [1,2,3,4,5]` will result in `[3,7,5]`.

5. **(10pt)** Write the function classAverages, which takes a list of students, represented by 4-tuples, and yields a list containing their overall average score for the class. Use the weights presented in the syllabus. Create as many helper functions as you need. You may beed the `fromIntegral` function, which maps an integer to another number type.

6. **(5pt)** Define a Haskell function (called removeVowels) that removes all vowels (Don't count y) from a string.

7. **(5pt)** Define a Haskell function (called reversedWords) that breaks a string into a list of words (consecutive non-whitespace characters) where each word is reversed.

Common Weakness Enumeration

8. **(2pts)** What is "Use After Free"? What are the two common causes?

9. **(2pts)** What is "Improper Input Validation"? How is it related to CWE-89 and CWE-78?

10. **(2pts)** What is "Integer Overflow or Wraparound". How can it affect security?

11. **(2pts)** Why is "Use of Hard-coded Credentials" considered a weakness?

12. **(2pts)** "Out-of-bounds Write" and "Out-of-bounds Read" are more specific versions of what common weakness?

13. **(2pts)** What is the "Uncontrolled Resource Consumption" weakness? What are the two common situation where incorrect implantation can lead to "Uncontrolled Resource Consumption"?

14. **(2pts)** What is a "NULL Pointer Dereference". Which major languages allow for programs that allow this weakness?

15. **(3pts)** What is Cross-site scripting? Briefly describe the three main types.

16. **(3pts)** What is the "Path Traversal" weakness? Briefly describe two ways it can be taken advantage of?

17. **(3pts)** Consider CWE-200, "Exposure of Sensitive Information to an Unauthorized Actor". Briefly explain Example 1 given on its webpage.

18. **(3pts)** What is CWE-352, "Cross-Site Request Forgery (CSRF)"? Give an example.

19. **(10pts)** Describe CWE-434, "Unrestricted Upload of File with Dangerous Type". Give a real world example of software that avoids this weakness. What could happen, in this example, if the software did not avoid this weakness?

20. **(10pts)** Consider the following code:

```
socket = waitforclient()
sText = readline(socket.in)
object = deserialize(sText)
result = object.doWork()
send(socket.out, result)
:
:
```

Which CWE does it have? What problems could this cause? Discuss how this program can be fixed so that it no longer has this weakness.