University of Calgary
Department of Computer Science

Introduction to Programming Paradigms (CPSC 449)
Assignment 1 of 4

This assignment is to be completed individually and you will only receive credit for your own original work.

Your submission will be evaluated with respect to style and efficiency, as well as effectiveness.

Reproducing or adapting any published or unpublished material, regardless of the source, without including a proper reference to the source is considered academic misconduct and may result in very severe penalties.

For this assignment you will be writing Haskell source code that can be used by the Glasgow Haskell Compiler (in interactive mode) to encrypt and decrypt Vigenère ciphers (http://en.wikipedia.org/wiki/Vigenere_cipher). You can assume that your programs are only expected for work with uppercase letters. You may not use any functions that you do not write yourself, except for the built-in `length` function and the `ord` and `chr` functions defined below (for converting between characters and there corresponding ASCII codes).

```
ord :: Char -> Int
ord c = fromEnum c
```

```
chr :: Int -> Char
chr i = (toEnum i)::Char
```

---

1. Write a function (with a type definition) that takes a single character argument and produces a single integer return value. If the character is an uppercase letter, this function should return an integer between 0 and 25 inclusively, corresponding to the index of that letter in the English alphabet. To clarify, calling the function with the argument "A" should result in the integer 0, calling the function with the argument "B" should result in the integer 1, etc. For any other input, the function should return the integer 25. You must use guards to write this function – do not use the if-then syntax.

2. Write a function (with a type definition) that takes two strings (i.e., lists of characters) k and p, and returns a string that is composed of the characters of k repeated until a string that is the same length as p has been created. To clarify, if your function was named foo, then foo "KEY" "MESSAGE" should return "KEYKEYK". (This is the approach by which the keyword of a Vigenère cipher is made compatible with the length of the message you wish to encrypt.)

3. Using the functions you have created for questions 1 and 2 above, write an encryption function (with a type definition) that takes two strings – a keyword and a plaintext message to be encrypted – and returns a string of text that has been encrypted with the Vigenère cipher corresponding to that keyword. Write a complementary decryption function that takes two strings – a keyword and a ciphertext (i.e. encrypted) message to be decrypted – and returns a string of text that has been decrypted. (This will, naturally, require that you research the Vigenère ciphers, and for that you are assigned to read the description that can be found at http://en.wikipedia.org/wiki/Vigenere_cipher#Description.)

4. If your encryption and decryption functions have been implemented correctly, you can make the assertion that the a decryption function call can be used to reverse an encryption function call (if applied with the same key, of course). Use this fact to design a property test that can be used (with Test.QuickCheck) to test your program. To accomplish this you will probably need to write a function that removes all the non-uppercase letters from a string (to ensure that the strings randomly generated by quickCheck are viable) – this is not the only approach but it is probably simplest.