**Function Programming**

**Oghenenohwo Ojakovo**

**Exercise 1**

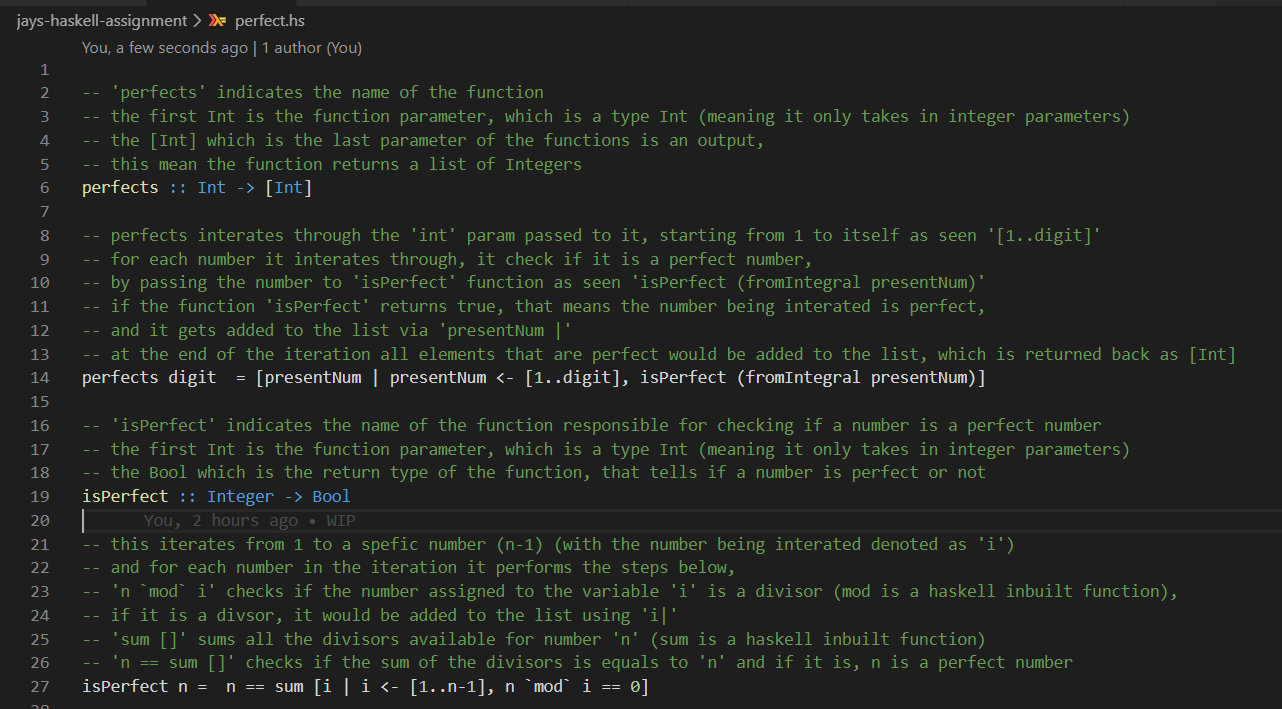
A positive integer is perfect if it equals the sum of all of its factors, excluding the number itself. Using a list comprehension, define a function

perfects:: Int -> [Int]

that returns the list of all perfect numbers up to a given limit. For example:

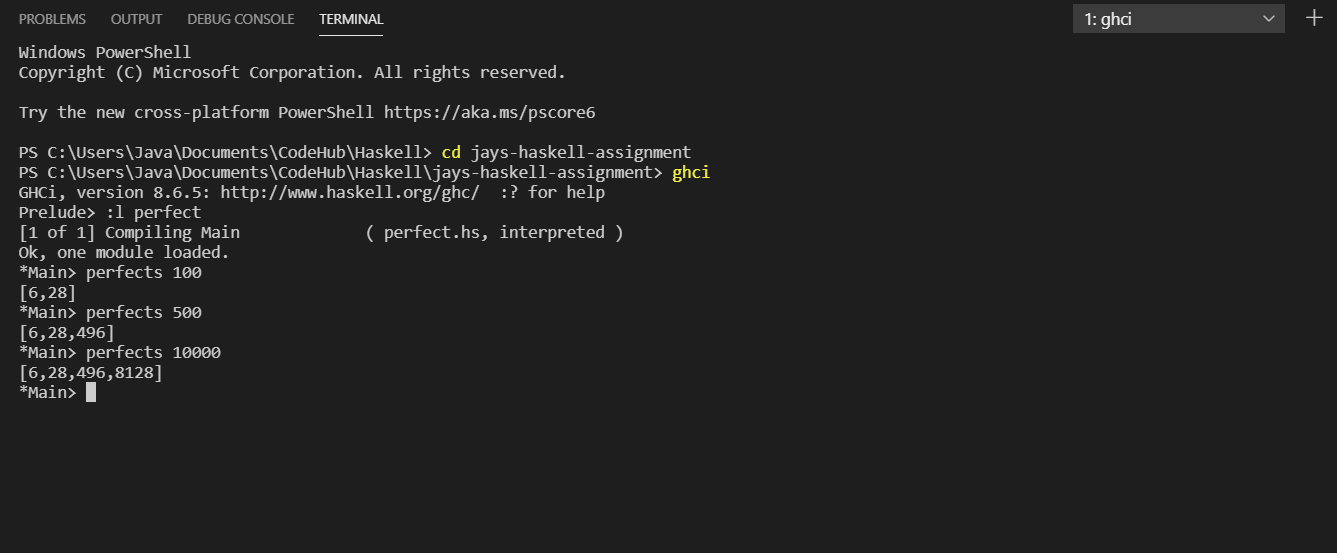
Ghci> perfects 500

[6,28,496]



A perfect number can be defined as a number which the sum of its divisors equals itself. The code above uses the aid of iteration, lists and conditions to aid it achieve that task.

Upon running the code with gchi: and typing “**perfects n**”, the code iterates through from one to the specific number (n-1). As seen below perfects 100 gives [6, 28], perfects 500 gives [6, 28, 496] while perfects 10000 gives [6, 28, 496, 8128]. Note as the number of ‘n’ increases this causes the time complexity of the algorithm to increase due to the number of times the function has to iterate, this would make the function slow.



**Exercise 2**

Write a Caesar Cipher program: The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials. Thus, to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down. The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

Examples :

Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ

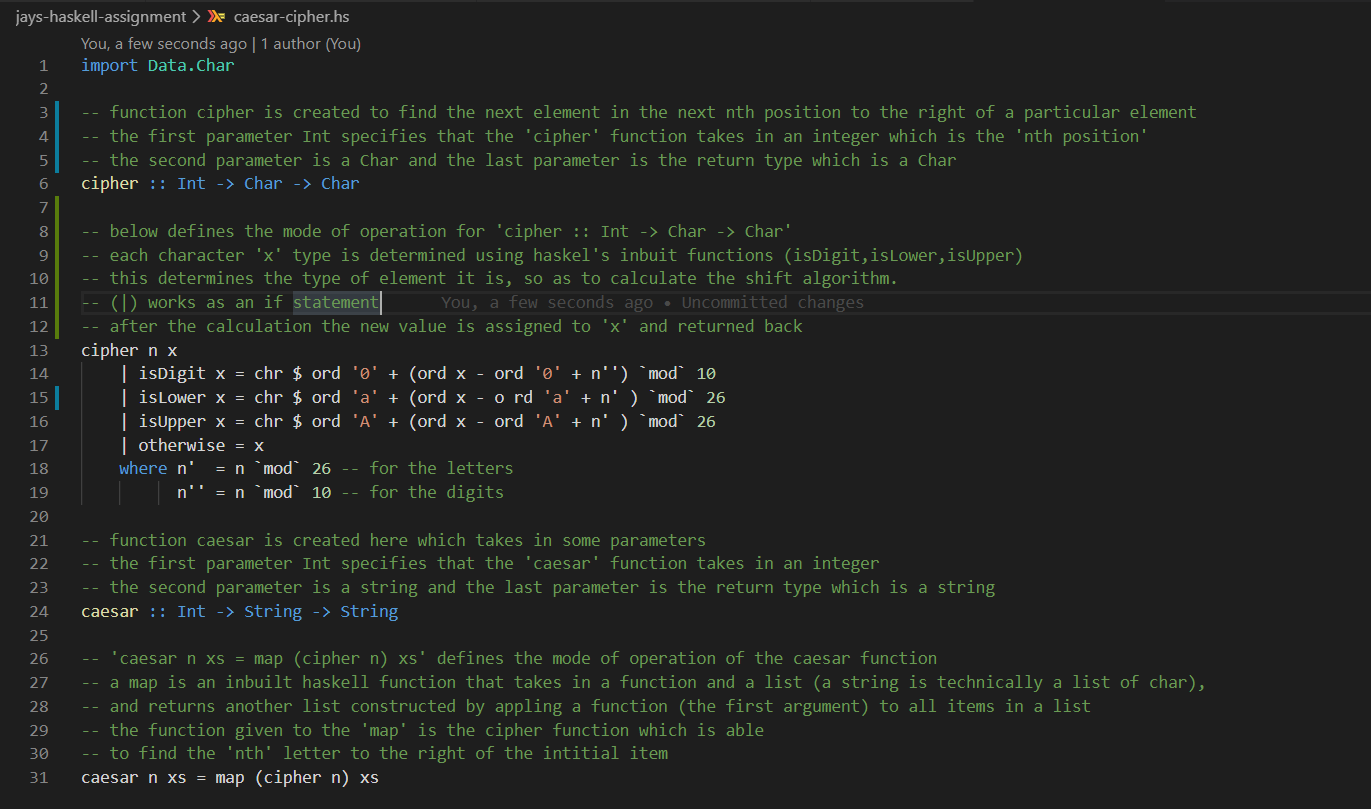
Shift: 23

Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW

Text : ATTACKATONCE

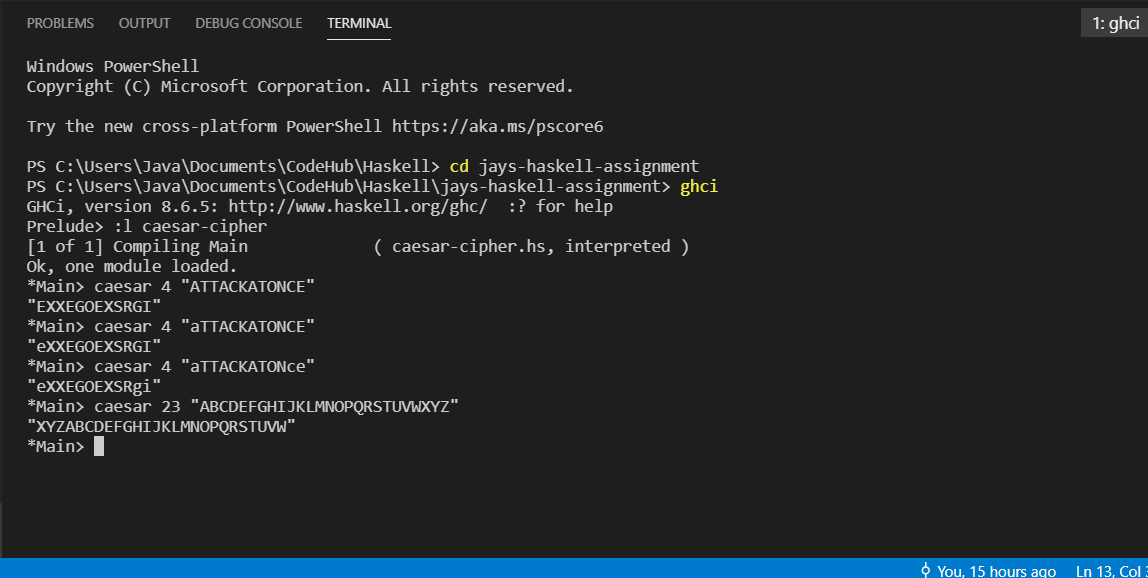
Shift: 4

Cipher: EXXEGOEXSRGI



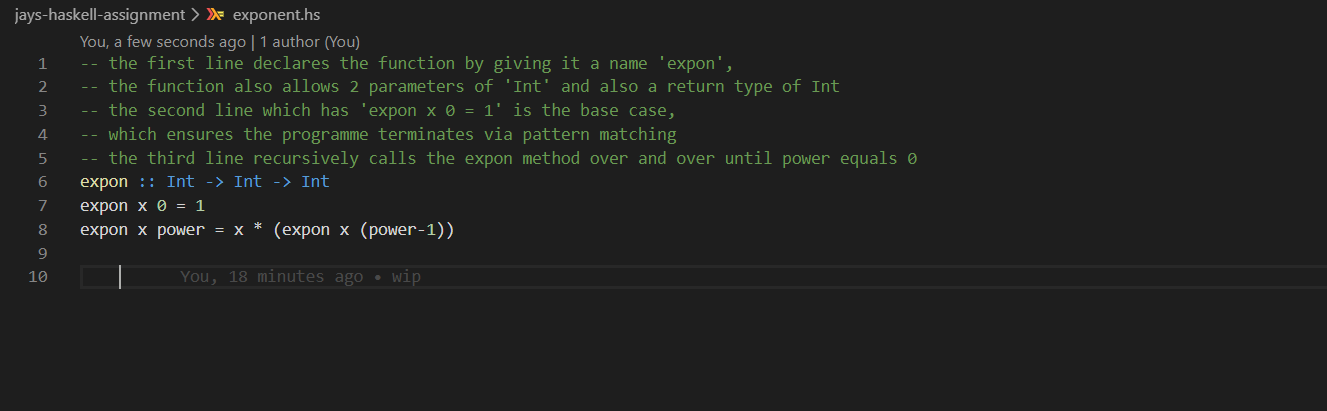
Implementing a Caesar cipher,is to encode the string which simply replaced each letter in the string by the letter that will places further down and wrapping around the alphabet until the end of the alphabet. However, the use Caesar cipher can easily cracked by exploiting information about letter frequencies in english text.

As you can see the output below in WinGHCi, the alphabet will encrypt every letter in the message and each letter of the plain text message will be replaced.

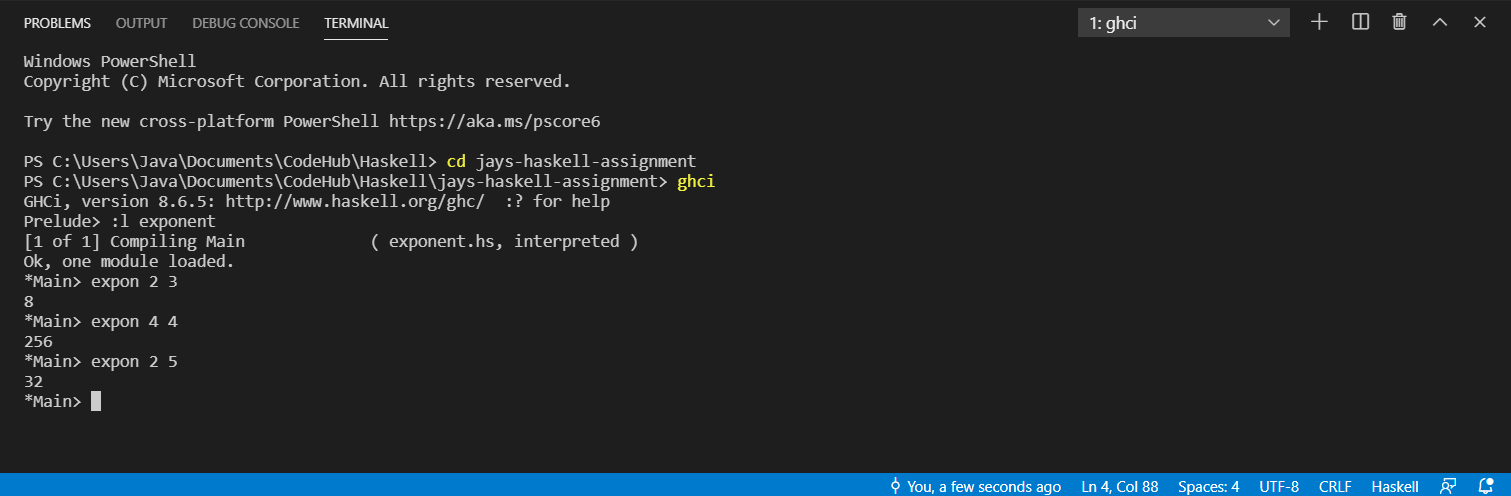


**Exercise 3**

Define the exponentiation operator ^ for non-negative integers using the same pattern of recursion as the multiplication operator \*, and show how 2 ^ 3 is evaluated using your definition.



here as you can see, it works properly as I entered 4^4, 2^5 and it shows the result.



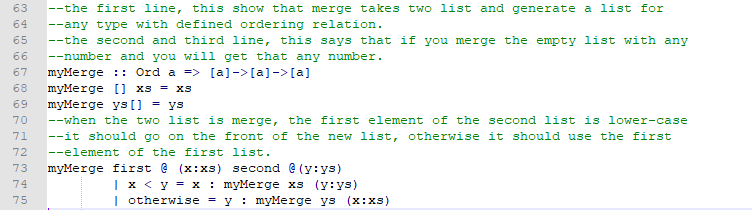
**Exercise 4**

Define a recursive function merge :: Ord a => [a] -> [a] -> [a] that merges two sorted lists to give a single sorted list.

For example: > merge [2,5,6] [1,3,4]

[1,2,3,4,5,6]

Note: your definition should not use other functions on sorted lists such as insert or isort, but should be defined using explicit recursion.



As you can see the output in WinGHCi,I manage the merge [2,3,4][5,6] and as a result, it will merge the number thus if the number is lower then it will be the first number on first list.

