SCHOOL OF ENGINEERING AND TECHNOLOGY

FINAL ASSESSMENT FOR THE BSC (HONS) COMPUTER SCIENCE; YEAR 2

ACADEMIC SESSION 2022; SEMESTER 3

PRG2214: Functional Programming Principles

Duration: 1 Week

Project **DEADLINE: Week 14**

INSTRUCTIONS TO CANDIDATES

- This assignment will contribute 50% to your final grade.
- This is an individual assignment.

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- Coursework submitted after the deadline but within 1 week will be accepted for a maximum mark of 40%.
- Work handed in following the extension of 1 week after the original deadline will be regarded as a non-submission and marked zero.

Lecturer's Remark (Use additional sheet if required)

I **Lee Jia Qian** (Name) **19117613** std. ID received the assignment and read the comments.

.....03/12/2022..... (Signature/date)

Academic Honesty Acknowledgement

"I **Lee Jia Qian** student name). verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment."

….....03/12/2022... (Student's signature / Date)

Table of content

BMI Calculator

1. Introduction

This is a Function Programming final project using Haskell language, which is a command-line application that simulates the use of the Body Mass Index (BMI) Calculator to let users calculate their BMI that falls into which category. This system takes the users' input of their weight and height either in the imperial unit (pounds/ inches) or metric unit (kilogram/ centimeter) and calculates their BMI then outputs the result.

2. Library and module

There is a use of a library from the Base library which I have imported a module from the Base package that is Text.Printf as can see in figure 1. From Text.Printf, I have utilized ***printf "%.4s"*** where the ***s*** is from String value then it will only display 4 String from the user input that have been converted from Double to String.

Figure 1

3. Data type and function

In Haskell language, there are many data types such as Bool, Int, etc. In this BMI Calculator system there have an own predefined data type using the data keywords.

Figure 2

From figure 2, the data type BMI that is used in the result function where statements will be displayed to the according to the bmi value.

```
instance Show BMI where
  show Underweight = "You are Underweight"
  show Normal = "You are Normal"
  show Overweight = "You are Overweight"
  show Obese = "You are Obese"
  show ExtremelyObese = "You are Extremely Obese"
```

Figure 3

The show typeclass instance shown in figure 3 was declared manually to ensure that each data constructor has its own String value to be printed when the console prints the data constructor.

```
metricBMI :: Fractional a => a -> a -> a
metricBMI weightKg heightCm = weightKg / (heightCm / 100)^2

imperialBMI :: Fractional a => a -> a -> a
imperialBMI weightLbs heightIn = (weightLbs * 703) / heightIn^2
```

Figure 4

The two functions from figure 4 are the metricBMI and imperialBMI function that will take 2 perimeters that are Fractional class. Both functions take 2 inputs from users which is their weight and height in either metric unit system using metricBMI function or in imperial unit system using imperialBMI with their own calculation format.

```
result :: (Ord a, Fractional a) => a -> a -> IO()
result x y  | bmi < 18.5 = print Underweight
            | bmi < 24.9 = print Normal
            | bmi < 29.9 = print Overweight
            | bmi < 34.9 = print Obese
            | otherwise = print ExtremelyObese
           where bmi = metricBMI x y
```

Figure 5

Figure 5 is the result function that takes in 2 inputs that are the Ordinal class since the comparison is used in the function to produce an output. The function uses guards to print a data constructor of type BMI. The calculation is done using the 2 inputs from the user and where the bmi value is calculated from the metricBMI function accepting a parameter of x and y. If the bmi value is lesser than a certain value, the respective data constructor will be printed. The data constructor can be printed out since Show instance has been declared. Note: This function is only for metric calculation

```
bmiChart :: [String]
bmiChart = ["Underweight     - < 18.5","Normal          - 18.5 ~ 24.9",
"Overweight      - 25 ~ 29.9", "Obese           - 30 ~ 34.9",
"Extremely Obese - > 35"]
```

Figure 6

In figure 6, bmiChart function is a list of String that store all the bmi category will print out in the console when user call the function.

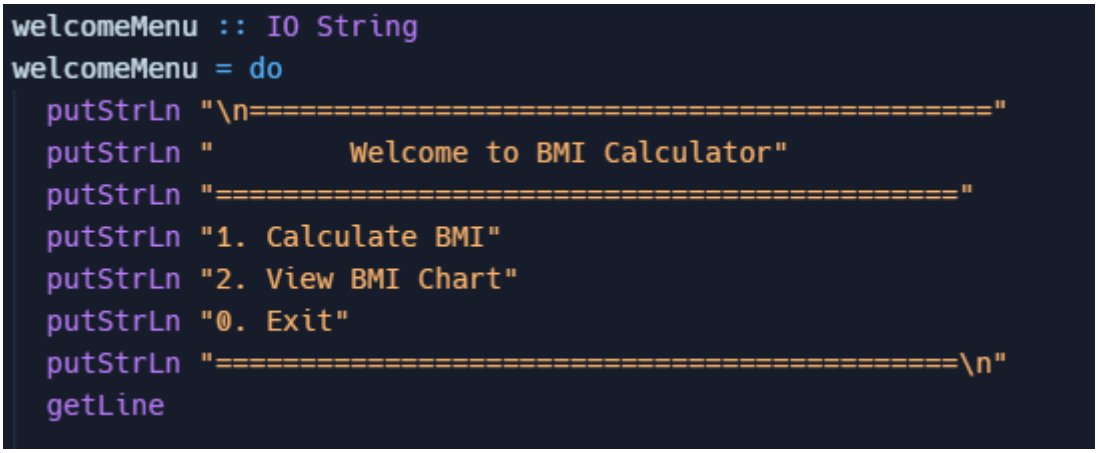

Figure 7

Figure 7 is the welcomeMenu function that prints out the welcome menu and selection option of the BMI calculator for user. The function will then receive a String input from user which is the selected option.

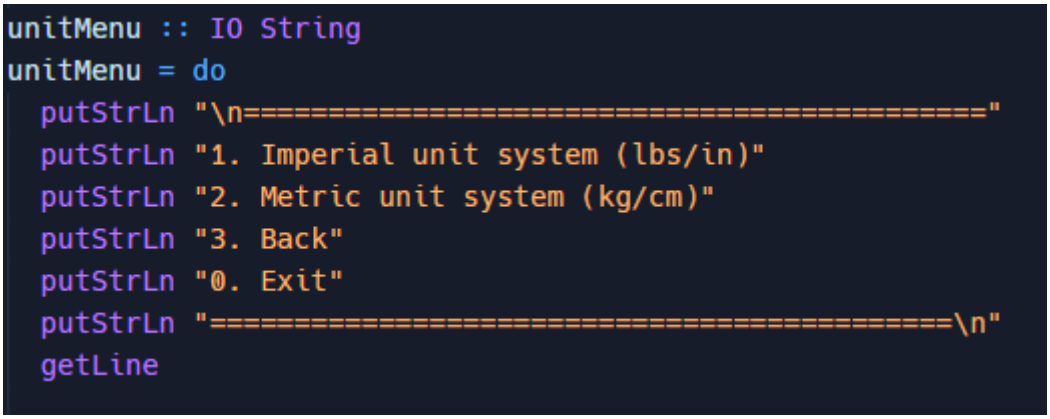

Figure 8

In figure 8, unitMenu function is printing the menu of the type of unit calculation for the bmi either in metric or imperial unit system. The function will then receive an input from user which a String that is entered by the user.

```haskell
continueMenu :: IO String
continueMenu = do
  putStrLn "\n=========================================="
  putStrLn "Do you wish to continue?"
  putStrLn "1. Continue"
  putStrLn "0. Exit"
  putStrLn "==========================================\n"
  getLine
```

Figure 9

From figure 9, continueMenu function is also a menu that prints the continue option after finish processing calculation of the bmi and displayed the result. The function will receive a String which is the selected option of user.

```haskell
welcomeSelection :: String -> IO ()
welcomeSelection "1" = unitMenu >>= unitSelection
welcomeSelection "2" = do
  mapM_ print bmiChart
  welcomeMenu >>= welcomeSelection

welcomeSelection "0" = putStrLn "Program ends..."
welcomeSelection _ = putStrLn "Input Error" >> welcomeMenu >>= welcomeSelection
```

Figure 10

In figure 10, welcomeSelection is a function that receive a String and return an IO (). welcomeSelection function let user select option after the welcomeMenu function (Figure 7) is printed by entering the number to proceed to next step. When option 1 is entered, the program will print the unit menu using the unitMenu function (Figure 8) . When user enter option 2, it will print each element from the list, bmiChart function using the mapM_ keyword, then will print the welcome menu to user again after display the chart. When users selected option 0, let users to exit the program and a program terminate message will be prompted to users. If an invalid option is entered by users, the program will prompt users to reenter a valid option by printing the welcome menu again.

```haskell
unitSelection :: String -> IO ()
unitSelection "1" = do
  putStrLn "\n========================================="
  putStrLn "Please enter your weight in pounds(lbs)"
  weightString <- getLine
  let weight = read weightString :: Double
  putStrLn "Please enter your height in inches(in)"
  heightString <- getLine
  let height = read heightString :: Double
  let bmi = imperialBMI weight height
  let bmiString = show bmi
  putStrLn $ "Your weight is " ++ weightString ++ "lbs and height is " ++
heightString ++ "in and your BMI is " ++ printf "%.4s" bmiString

  if bmi < 18.5
    then putStrLn "You are Underweight"
  else if bmi >= 18.5 && bmi <= 24.9
    then putStrLn "You are Normal"
  else if bmi >= 25 && bmi <= 29.9
    then putStrLn "You are Overweight"
  else if bmi >= 30 && bmi <= 34.9
    then putStrLn "You are Obesity"
  else
    putStrLn "You are Extereme Obesity"

  continueMenu >>= continueSelection
```

Figure 11

In figure 11, unitSelection function is a function that receive a String and return an IO (). unitSelection function is used for users selected option from the unit menu (Figure 8). When user selected 1 in the unitMenu, it will print a statement to let user know what to input, then get input from user and calculate the bmi using imperialBMI function (Figure 4) calculation. After the calculation it will print the result and which bmi category the users fall into using the if else statement, then will prompt user whether they wanted to continue or not by using the continueMenu function (Figure 9).

```
unitSelection "2" = do
  putStrLn "\n========================================="
  putStrLn "Please enter your weight in kilogram(kg)"
  weightString <- getLine
  let weight = read weightString :: Double
  putStrLn "Please enter your height in centimeter(cm)"
  heightString <- getLine
  let height = read heightString :: Double
  let bmi = metricBMI weight height
  let bmiString = show bmi
  putStrLn $ "Your weight is " ++ weightString ++ "kg and height is " ++
heightString ++ "cm and your BMI is " ++ printf "%.4s" bmiString
  result weight height
  continueMenu >>= continueSelection
```

Figure 12

Figure 12 show when users enter 2 to choose option 2 which is a metric unit system that is in kilogram (kg) and centimeter (cm). Same as figure 11 above, it will receive 2 inputs from users after displayed what to enter from the following statement. After getting their weight and height in metric unit system then it will calculate the bmi using the metricBMI function (Figure 4) calculation. The main different from figure 11 is it take the weight and height input from user apply on the result function (Figure 5) to print the bmi category where the users fall into without the use of if else statement but using a function. After showing the result, the program will prompt user whether they wanted to continue or not by using the continueMenu function (Figure 9).

```
unitSelection "3" = welcomeMenu >>= welcomeSelection
unitSelection "0" = putStrLn "Program ends..."
unitSelection _ = putStrLn "Input Error" >> unitMenu >>= unitSelection
```

Figure 13

In figure 13, when users entered 3 it will bring users back to the welcome menu, while 0 is entered the program will be terminated printing “program ends …” message to users. If an invalid option is entered by users, the program will prompt users to reenter a valid option by printing the unit menu again.

```
continueSelection :: String -> IO()
continueSelection "1" = unitMenu >>= unitSelection
continueSelection "0" = putStrLn "Program ends..."
continueSelection _= putStrLn "Input Error" >> continueMenu >>=
continueSelection
```

Figure 14

Figure 14 shows the continueSelection function which receive a String and return an IO (). continueSelection function let users select the option to continue after the program finish running and printed the result. When option 1 entered it will bring users back to the unit menu to choose their desire unit system to calculate their bmi from the unitMenu function (Figure 8). When 0 is selected by users, the program will be ended and exit by prompting termination message to users. If an invalid option besides 1 and 0 is entered by users, the program will prompt users to reenter a valid option by printing the continue menu again.

```
main :: IO()
main = welcomeMenu >>= welcomeSelection
```

Figure 15

From figure 15 is the main function of the system that receive IO (). This function call the welcomeMenu function (Figure 7) to print out the welcome menu to users and the follow the step shown in the menu so users can proceed to next step to calculate their bmi value. By calling this function in the console to run the program.

4. Working Use Cases

```
> main

==========================================
        Welcome to BMI Calculator
==========================================
1. Calculate BMI
2. View BMI Chart
0. Exit
==========================================
```

Figure 16

Figure 16 is the welcome menu of the program where users will be required to enter their option. Where option 1 is calculate the bmi, option 2 is view the bmi chart and option 3 is exit the program.

```
==========================================
        Welcome to BMI Calculator
==========================================
1. Calculate BMI
2. View BMI Chart
0. Exit
==========================================

2
"Underweight     - < 18.5"
"Normal          - 18.5 ~ 24.9"
"Overweight      - 25 ~ 29.9"
"Obese           - 30 ~ 34.9"
"Extremely Obese - > 35"
```

Figure 17

Figure 17 shows after users enter 2 as the option it will print the bmi chart to the user.

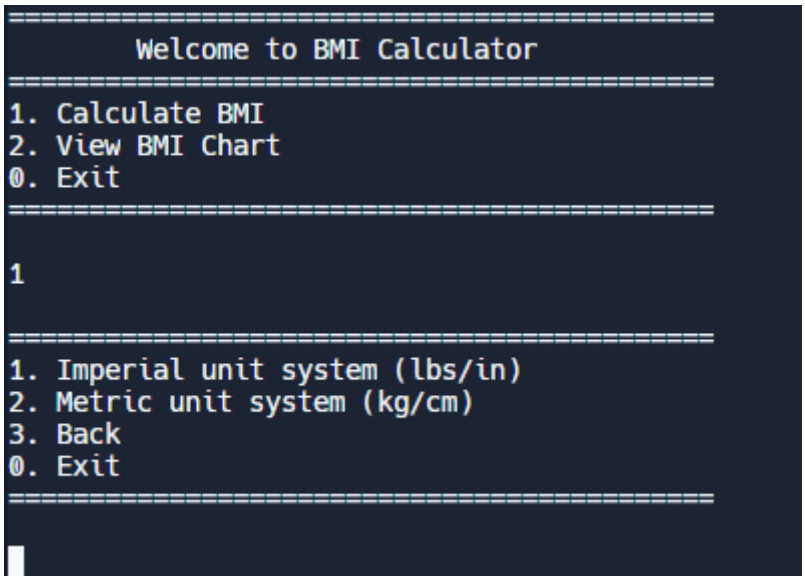

Figure 18

In figure 18 shows when users enter 1 as the option it will bring users to another menu page which let user to choose from either 1 for imperial or 2 for metric unit system. Users select 3 will bring them back to welcome menu and 0 is to exit the program.

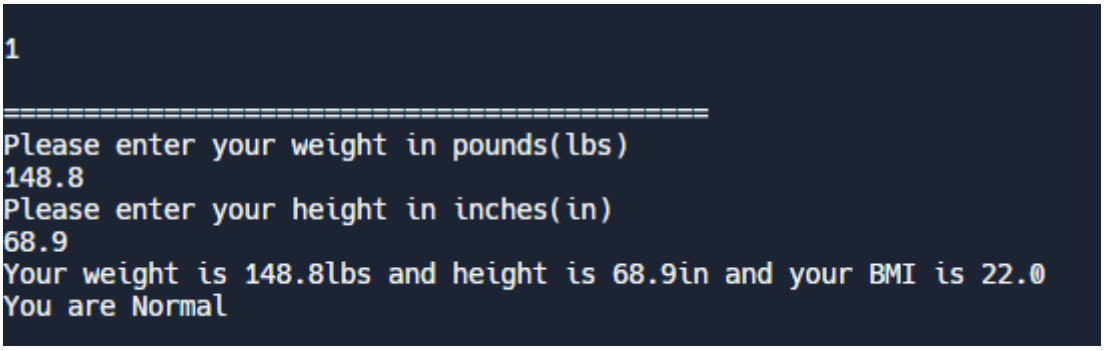

Figure 19

Figure 19 shows when choose imperial unit system which is 1, then the users must enter their weight in pounds and height in inches. Then the system will print the weight and height they entered and the calculated bmi value along with the bmi category with is normal in this case. The same process will happen if users choose option 2, users will then have to enter weight in kilogram and height in centimeter instead of pounds and inches. Following the weight, height along with the bmi value and the bmi category will also be printed out same as the option 1.

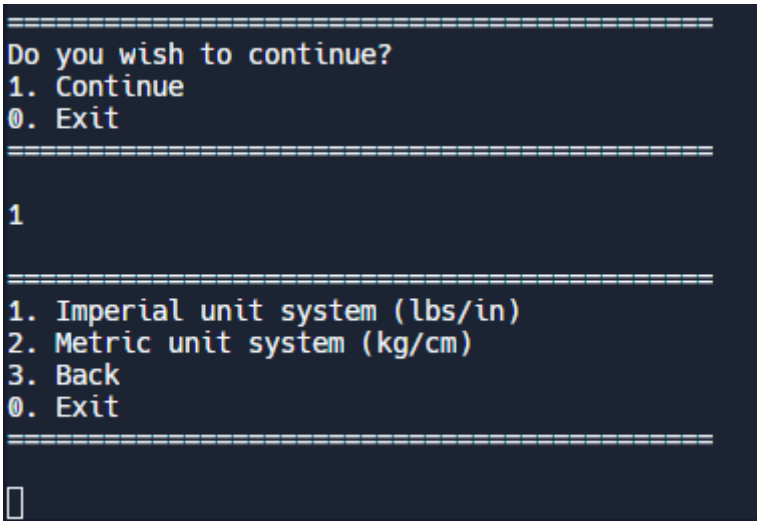

Figure 20

In figure 20, after the program printed the result for the users it will print the continue menu to users asking them whether they wanted to continue for 1 is entered or exit the program for 0 is selected. When option 1 is entered it will bring users back the unit menu to choose the unit system wanted to calculate the bmi.

5. Personal Reflection

The use of functional programming concept in this final project assignment can be expressed using type constructors, data constructor and typeclass instance along with a library and module. The Haskell BMI Calculator program have used the basic of functional programming concept that have been taught in the lecture and practical session. The textbook I have referred to is “Haskell Programming from first principles” by Christopher Allen and Julie Moronuki. Other than that, I have also referred to the practical session exercise and online research is done to learn the use of if else statement and the Hoogle for importing libraries and modules.

The biggest issue I ran into when developing this application was the inability to store and remove previous data from a list, so if a user for example a doctor wanted to check on his/her patient bmi can be retrieved easily or remove the previous patient bmi and replaced it with a new one. On the other hand, the other problem I have encountered which is the lack of time in creating a better program. It is a challenging to schedule a suitable time to complete this project as soon as possible because there are final exams for other subjects the following week. Due to the anticipated lack of time, I quickly decided on the topic I wanted, did some research and refer from the recommended textbook, online sources and practical exercises.

For strength of this program is that the BMI calculator has a clean, easy and straightforward interface and information for users to use the system. The program is easy to use as it only requires a few inputs from users to get the result there are asking for. In terms of the weaknesses, there are no error handling features implemented in some part of the program. When users entering their weight and height that only accept numeric value, if user input a string value instead of numeric value, the program then will capture the error and throw an error message to user asking them to reenter.

Program link: https://replit.com/@19117613/final-assignment-bmi-calc#Main.hs
Demo link: https://youtu.be/gn0_d1JIg4A