Specification for Assignment 2 of 4

Your submission for this assignment ***must include your full name*** *(as it appears on cuLearn) and you nine-digit* ***student number*** *at the top of* ***every file you submit. Source code submissions that crash*** *(i.e., terminate with an error) on execution will* ***receive a mark of 0.***

Your submission for the first question must be a ***single source file*** *with a* ***file name of 'comp3007_w19_#########_a2_1.hs'*** *(with the number signs replaced by your nine-digit student number). It* ***must be written using Haskell*** *and* ***must run in GHCi or WinGHCi.***

Your submission for the second question must be a ***single source file*** *with a* ***file name of 'comp3007_w19_#########_a2_2.hs'*** *(with the number signs replaced by your nine-digit student number). It* ***must be written using Haskell*** *and* ***must run in GHCi or WinGHCi.***

Compress your submissions *into a* ***zip*** *file named* ***'comp3007_w19_#########_a2.zip'***

Late assignments will not be accepted *and will* ***receive a mark of 0.***

The due date for this assignment is Saturday, February 16, 2019, by 11:00pm.

Without exploring the IO monads in any great detail, the following block of code (provided to you in a support file) will allow you to open a bitmap as a two-dimensional list of three-tuples of Ints (for the red, green, and blue component values of each pixel in the bitmap) and two additional Ints for storing the dimensions (width and height) of the image.

```
loadBitmap :: FilePath -> [[(Int, Int, Int)]]
loadBitmap filename = repackAs2DList (either returnEmptyOnError processDataOnBMP (unsafePerformIO (readBMP
filename)))

returnEmptyOnError :: Error -> ([(Int, Int, Int)], (Int, Int))
returnEmptyOnError _ = ([], (0, 0))

processDataOnBMP :: BMP -> ([(Int, Int, Int)], (Int, Int))
processDataOnBMP bmp = ((parseIntoRGBVals (convertToInts (unpack (unpackBMPToRGBA32 bmp)))), (bmpDimensions
bmp))

convertToInts :: [Word8] -> [Int]
convertToInts [] = []
convertToInts (h:t) = (fromIntegral (toInteger h)) : (convertToInts t)

parseIntoRGBVals :: [Int] -> [(Int, Int, Int)]
parseIntoRGBVals [] = []
parseIntoRGBVals (h:i:j:_:t) = (h,i,j) : (parseIntoRGBVals t)

repackAs2DList :: ([(Int, Int, Int)], (Int, Int)) -> [[(Int, Int, Int)]]
repackAs2DList (pixels, (width, height)) = (Prelude.reverse (repackAs2DList' pixels width height))

repackAs2DList' :: [(Int, Int, Int)] -> Int -> Int -> [[(Int, Int, Int)]]
repackAs2DList' []  width  height = []
repackAs2DList' pixels width height = (Prelude.take width pixels) : (repackAs2DList' (Prelude.drop width
pixels) width height)
```

To complete this assignment, you will need to install the package "bmp-1.2.6.3". Consult https://wiki.haskell.org/Cabal/How_to_install_a_Cabal_package for additional information and check the comments at the top of the source file provided.

Question 1: "Custom Palette ASCII Art" (10.0 marks)

For the first question of this assignment, you will design and write a program in Haskell that will display an image as ASCII art, using the first argument provided as though it were a grayscale palette. Your function will also take a single Boolean value as a second argument, to determine whether or not the palette should be used from left-to-right or right-to-left (the former being used for black text on a white background and the latter being used for white text on a black background. The third argument will be the filename to be loaded.

The provided `loadBitmap` function, when called using:

```
loadBitmap "sample_image_to_search.bmp"
```

would result in a two-dimensional list of pixels as a return value (with each pixel being a three-tuple of values between 0 and 255 inclusive). This means that, if you were to name your function `foo`, the function call:

```
foo ".-+*#" True (loadBitmap "sample_image_to_search.bmp"))
```

would load the image and convert it such that "`.`" would be used for white pixels, "#" would be used for black pixels, and the characters "`-+*`" would be used for three different shades of grey - 75% grey (#BFBFBF), 50% grey (#7F7F7F), and 25% grey (#3F3F3F), respectively. n.b., If the second argument had been False, then "#" would have been used for white, "`.`" for black, etc., and had the first argument been a longer string, then more "shades of grey" would have been used.

The `showAsASCIIArt` function (also provided), will allow you to view the result of your function, using a call such as the following:

```
showAsASCIIArt (foo ".-+*#" True (loadBitmap "sample_image_to_search.bmp")))
```

With the exception of the `fromIntegral` and `round` functions, you may not use any built-in functions to complete this assignment. This means that you will need to write your own implementations of basic functions like `length`, `take`, `drop`, and `reverse`, should you required them. You can, however, use the list indexing operator (`!!`), the list constructor operator (`:`), and any arithmetic operators you wish.

You should approach this problem in stages, beginning first with some research on the topic of luminosity so that you understand how best to determine what shade of grey is best suited for a specific pixel. Then you should be able to write recursive functions (n.b., you will require at least two) for processing a two-dimensional list of integers into a two-dimensional list of characters.

Question 2: "Where's Waldo" (10.0 marks)

For the second question of this assignment, you will design and write a program in Haskell that will load two bitmap images, convert them to ASCII art using the code you wrote for question 1, and then determine whether or not the second image appears in the first one and, if it does, provide the co-ordinates.

As a clarifying example, the `sample_image_to_search.bmp` file provided could be represented in ASCII art as follows:

Were you to also load "`sample_image_to_find.bmp`" and convert that to ASCII art using the same palette, then if you were to search the former image for the latter image your program should report that the "`sample_image_to_find.bmp`" was found at (252, 107) in "`sample_image_to_search.bmp`". If you were to name this function `bar` (and were still using the name `foo` for the function you wrote for question 1), you would call your searching function using the following syntax.

```
bar (foo " .:-=+*#%@" True (loadBitmap "sample_image_to_search.bmp")) (foo " .:-=+*#%@" True (loadBitmap "sample_image_to_find.bmp"))
```

n.b., You are expected to use meaningful function names in your submission. Do not use the names foo and bar. Furthermore, as was the case in the previous question, with the exception of the `fromIntegral` and `round` functions, you may not use any built-in functions to complete this assignment

The programs you write must be a completely original works, authored by you and you alone, prepared for this offering (i.e., Winter 2019) of COMP3007. Do not discuss this (or any other) question with anyone except the instructor or the teaching assistants, and do not copy materials from the internet or any other source.