Your submission for this assignment ***must include your full name*** *(as it appears on cuLearn) and you nine-digit* ***student number as a comment*** *at the top of* ***every source file you submit****.*

Your submission for question 1 must be a ***single source file*** *with a* ***file name of 'comp3007_f18_#########_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 question 2 must be a ***single source file*** *with a* ***file name of 'comp3007_f18_#########_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 archive named* ***'comp3007_f18_#########_a2.zip'***

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

Submissions that crash *(i.e., terminate with an error) on execution will* ***receive a mark of 0****.*

The due date for this assignment is Saturday, October 13, 2018, by 11:00pm.

Question 1: "Dirty QR Codes" (12.0 marks)

For the first question of this assignment, you will design and write a program in Haskell that will read, process, and display (as ASCII art) a simple matrix barcode.

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 3-tuple containing a list of 3-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.

```
loadBitmapIntoIt :: FilePath -> IO ([(Int, Int, Int)], Int, Int)
loadBitmapIntoIt filename = do
  (Right bmp) <- readBMP filename
  return ((parseIntoRGBVals (convertToIntList (unpack (unpackBMPToRGBA32 bmp)))), (fst (bmpDimensions
bmp)), (snd (bmpDimensions bmp)))

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

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

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.

When GHC is running in interactive mode (i.e., GHCi or WinGHCi), a function call to:

```
loadBitmapIntoIt <filename>
```

will load the bitmap image and return it, and since GHCi will bind the result of the last expression you have evaluated to the name "it", immediately after loading the bitmap (using the syntax above) it is possible to treat "it" as though it were a variable. The following syntax, for instance, will describe the type of "it":

```
:t it
```

If the previous statement is executed in GHCi immediately after a bitmap was successfully loaded, the interface will report the following:

```
it :: ([(Int, Int, Int)], Int, Int)
```

In other words, it was loaded as a 3-tuple of a list of 3-tuples of Ints, an Int, and another Int.

Once you have renamed the provided support file (using the convention specified above) and added comments with your name and student number to the top of that file, you will need to complete the following tasks:

a) You must write a recursive function that takes something of the form:

```
([(Int, Int, Int)], Int, Int)
```

as an argument and returns something of the form:

```
[[(Int, Int, Int)]]
```

which should be interpreted as a list of lists of 3-tuples of RGB values. You may use helper functions if you wish, but your solution must be recursive. You must document the name of your function at the beginning of your code using a comment like

```
-- foo :: ([(Int, Int, Int)], Int, Int) -> [[(Int, Int, Int)]]
-- foo is the function I have written to address requirement (a)
```

b) If you open one of the sample QR-codes in a drawing program you will notice that the image contains some monochromatic "noise" (i.e., the image isn't exactly "black and white"). You must write a recursive function to clean this data such that tuples in the list of list of 3-tuples are only either black (0, 0, 0) or white (255, 255, 255). You may again use helper functions if you wish, and you must again document the name of your function at the beginning of your code using a comment.

c) You must write a function:

```
showAsASCIIArt' :: [[(Int, Int, Int)]] -> [[Char]]
```

↑ *please note the apostrophe*

that will create an ASCII art representation of the loaded image as a list of lists of Chars. You may use whatever characters you wish, provided that there is a distinct contrast between the colours you use for black and white, respectively. In the support file you will note that the function

```
showAsASCIIArt :: [[(Int, Int, Int)]] -> IO ()
```

has been provided. This function calls `showAsASCIIArt'`, so if you are testing your solutions to requirements (a) and (b) above, you will need to "comment out" `showAsASCIIArt` until you have written `showAsASCIIArt'`.

Please note that, immediately following:

```
loadBitmapIntoIt <filename>
```

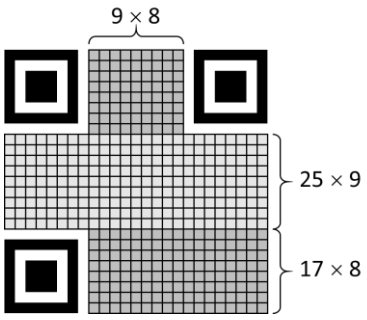
with a function call to:

```
showAsASCIIArt (<what_you_called_b> (<what_you_called_a> it))
```

should show you an image that matches the bitmap in `<filename>`. Pay special attention to ensure that the image is not flipped or rotated.

If you wish to receive a bonus mark for this assignment, you must complete the following additional task:

d) Although a genuine QR code contains much more than simply encoded data and position detection patterns (those 7×7 regions in the top-left, top-right, and bottom-left corners), for the final task you can assume that the remaining regions of the QR code (once cleared of "noise") contain only individual (and unmasked) bits – a white pixel is used to indicate a zero and a black pixel is used to indicate a one. Since the dimensions of the QR codes you will be working with are always 25 × 25, this means that the QR code can contain 433 bits (i.e., just over 54 bytes) of data. (n.b., The image above is included to clarify how the 433 bit total was calculated.)



For the sample QR codes you will be provided, these 54 bytes will represent a string of 54 ASCII characters. For the bonus, you must write a function that returns the 54-character string when passed the return from your functions for (b) and (a) above. This function must be recursive but you may use helper functions if you wish. you must again document the name of your function at the beginning of your code using a comment.

Please note that, with the exception of the `take` and `drop` functions that can be found in the Prelude library, you must write every function you use for this assignment yourself (although you are, of course, free to use built-in operators like +, –, &&, ||, :, ++, etc.)

Question 2: "Three-Valued Logic" (8.0 marks)

Three-valued logic is a type of many-valued logic that is defined for three distinct truth values - True, False, and Unknown. This system is of particular interest in the field of database administration as it facilitates logical calculations between the standard (i.e., binary) truth values and the NULL value that is frequently stored in an attribute. For the second exercise of this assignment you are required to define an enumerated algebraic data type (and some support functions) to be used to represent three-valued logic.

You MUST do some very basic research on three-valued logic before attempting this question. If you do not, then you will not know how the operators are expected to "behave".

In a separate file (i.e., separate from your submission for the first exercise) that you have named using the convention specified above and commented with your name and student number, you will need to complete the following tasks:

a) Define an enumerated algebraic data type to reflect the different possible values. I strongly recommend that you do not attempt to overload keywords like true and false; instead you should use troo, falz, and unknown (or something to that effect).

b) Write functions ternaryNOT, ternaryAND, and ternaryOR, defined for your enumerated type. Use the minimum number of patterns required - do not replicate entire truth tables or you will be penalized.

Both programs you submit for this assignment must be a completely original works, authored by you and you alone, prepared for this offering (i.e., Fall 2018) 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.