CS316, 2014 Worksheet 4. Higher Order Functions

Some of the follwing functions have types which use type classes which are not covered in this course. Consequently, I have supplied the types for these functions, eg the type of `insertBy` is

```
Ord b => (a -> b) -> a -> [a] -> [a]
```

You should ignore the `Ord b` part of such type definitions. If you really want to know about type classes, ask me or one of the other demonstrators.

Download the file Prac4.hs and edit the file to contain your answers. At the top of the file, write your name and user name in the space provided. Do not change the name of the file.

Question 1: Define the function `add` which takes a function `f` and a positive integer `n` as input, and returns

```
(f 1) + (f 2) + ...+ (f n)
```

Next define a function `allEqual` which again takes a function `f` and a positive integer `n` as input and returns `True` iff `f 1` up to `f n` are all equal.

Question 2: The function `fold` is defined as follows

```
fold f a []  =  []
fold f a (x:xs)  =  fold f (f a x) xs
```

Use `fold` to define the following functions:

- `fac` which takes an integer as input and returns the factorial of the input
- `maxList` which takes a list of positive integers and works out the maximum element of the list

Question 3: We can represent positive integers by list of integers, eg the integer `173` is represented by the list `[1,7,3]`. Use `fold` from question 2 to define a function `numeral` which takes a list of integers as input and returns the integer it represents.

Define a function `numeral2` which does the opposite of `numeral`. That is, `numeral2` takes a positive integer and returns its representation as a list of digits.

Question 4: Define a higher order version of the insertion sort algorithm. That is, define functions

```
insertBy   :: Ord b => (a -> b) ->  a  -> [a] -> [a]
inssortBy  :: Ord b => (a -> b) -> [a] -> [a]
```

such that `inssortBy f l` sorts the list `l` such that an element `x` comes before an element `y` if `f x < f y`.

Students sit three examinations and their results are represented by the type synonyms

```
type Result = (Mark,Mark,Mark)
type Mark = Int
```

Use `inssortBy` to answer the following questions. In testing your functions, you may wish to use the list of results `g500` which is defined in the file Prac4.hs.

Question 5: Define a function `sortTotal` which sorts a list of results such that the result with the highest total score is first in the resulting list.

Question 6: Define a function `sortbyModule` which takes an integer `i` and a list of results as input and sorts the list of results according to marks in the i'th module. Again, higher marks come first in the list

Question 7: A result is a *pass* if each of the three marks in the result is 40 or more, otherwise the result is a *fail*. Define a function `fl :: Result -> Bool` which returns `True` if the result is a fail.

Define a function `sortCut` which sorts a list of results according to the following criteria

- Any result which is a pass comes before any result which is not a pass.
- If two results are both passes, then the result with the higher total mark comes first.
- If two results are both fails, then the result with the higher total mark comes first.

Hint: Sortby a function which returns a pair type and investigate the order on pair types. For example is `(5,4)>(4,6)?`