Haskell Review 2

1. Write a function `minList :: [Integer] -> Integer` which returns the minimum element of a list.

   If the list is empty, you should return 0.

2. Write a function `addAbs :: [Integer] -> Integer` which adds together *the absolute value* of all the elements of a list.

   ```
   addAbs [1,-2,3,-4] = 10
   addAbs [1,10,-100] = 111
   ```

3. Write a function `existsOdd :: [Integer] -> Bool` which returns `True` if there exists an odd element in the list, and returns `False` otherwise.

4. Write a function `findOdd :: [Integer] -> Maybe Integer` which returns `Just x` if there is some `x` in the input that is odd, and returns `Nothing` otherwise.

5. Write a function `removeEmpty :: [String] -> [String]` which takes a list of strings and removes all empty strings from the list.
   `removeEmpty ["Hello","","there","","","world",""] = ["Hello","there","world"]`

6. Write a function `subtractEach :: [(Integer,Integer)] -> [Integer]` which takes a list of *pairs* of integers, and outputs a list consisting of their *differences*:
   `subtractEach [(5,1),(6,5),(7,10),(8,15),(9,20)] = [4,1,-3,-7,-11]`

7. Write a function `makeGreeting :: Maybe String -> String` which takes a string `name` wrapped in a `Maybe` type, and outputs "Hello, name!".
   If no name is provided, it should output "Hello!"
   ```
   makeGreeting (Just "Jesse") = "Hello, Jesse!"
   makeGreeting Nothing = "Hello!"
   ```

8. Write a function `catMaybes :: [Maybe a] -> [a]` which collects all non-`Nothing` values in the given list and puts them into a list of type `[a]`.
   `catMaybes [Nothing, Just 1, Just 5, Nothing, Nothing, Just 2] = [1,5,2]`

9. Write a function `classify :: [Either a b] -> ([a],[b])` which takes a list of "Either" type and returns a pair of lists. All `Left` values should go into the first list in the pair, and all `Right` value should go into the second:
   ```
   classify [Right 3, Left "hi", Left "there", Right 10, Right 4] = (["hi", "there"], [3,10,4])
   classify [Left 1, Right 'y', Left 5, Left 8, Right 'e', Right 's'] = ([1,5,8],"yes")
   ```

10. Write a function `isPrefix :: (Eq a) => [a] -> [a] -> Bool` and returns true if the first list is a prefix of the second list, and returns False otherwise.
    ```
    isPrefix "hey" "hello" = False
    isPrefix [1,2] [1,2,3,4,5] = True
    ```

Submission instructions

- Save your file as `LASTNAMErev2.hs`. Submit it to asulearn by the deadline.
- *Your file must compile/load correctly.*

  If your file does not load, gives a type error or some formatting error, you will not receive any credit for this assignment.
- *You should provide a definition for every function in the assignment.*

  If you can't get a function to work, use the `error` construct to complete the answer to that problem. For example,

```
collatz :: Integer -> Integer
collatz n = error "not done"
```