CS3200: Programming Languages
Homework 7

Required Problems

1. Write a funciton `nextToLast :: [a] -> a` which returns the element that is second to last in the list. If the list is too short, give an appropriate error message.

2. Write a function `mutiplyAndReverse :: (Num a) => a -> [a] -> [a]` which takes a number and a list of numbers as input and returns a list where each element of the list is multiplied by the input number and then the list is reversed. For example:

   ```
   Main> multiplyAndReverse 5 [] = []
   Main> multiplyAndReverse 5 [3,1] = [5,15]
   Main> multiplyAndReverse 2 [4,3,2,1] = [2,4,6,8]
   ```

3. Write a function `duplicates :: Eq a => [a] -> Bool` which returns True if its argument contains duplicate elements. For example:

   ```
   Main> duplicates [1,2,3,4,5]
   False
   Main> duplicates [1,2,3,2]
   True
   ```

4. Write a function `commaSeparate :: [String] -> String` that takes a list of strings and returns a single string that contains the given strings in the order given, separated by ", ". For example:

   ```
   commaSeparate [] = ""
   commaSeparate ["a", "b"] = "a, b"
   commaSeparate ["Monday", "Tuesday", "Wednesday", "Thursday" ]
            = "Monday, Tuesday, Wednesday, Thursday"
   ```

5. Write a function `deleteAll :: (Eq a) => a -> ([a] -> [a])` that takes an item (of a type that is an instance of the Eq class) and a list, and returns a list just like the argument list, but with the each occurrence of the item (if any) removed. For example.

   ```
   deleteAll 1 [1, 2, 3, 2, 1, 2, 3, 2, 1] = [2, 3, 2, 2, 3, 2]
   deleteAll 4 [1, 2, 3, 2, 1, 2, 3, 2, 1] = [1, 2, 3, 2, 1, 2, 3, 2, 1]
   deleteAll 3 [1, 2, 3] = [1, 2]
   ```

6. Write a function `makePalindrome :: [a] -> [a]` that takes as input a list, and turns that list into a palindrome by duplicating all the elements in the list. For example:

```
makePalindrome [1,2,3] = [1,2,3,3,2,1]
makePalindrome [‘a‘,‘z’,‘e‘] = [‘a‘,‘z‘,‘e‘,‘e‘,‘z‘,‘a‘]
makePalindrome [] = []
```

7. Extra credit: Write a function that takes a list as input and returns a sorted list, using any sorting algorithm EXCEPT quick sort, since we saw that one in class. (You also shouldn't use the built in sort! The point is to code it yourself.)