

CS3299: Programming Languages Homework 7

Required Problems

1. (a) Define a function `addFirstA` which takes a list of integers and returns a list in which each element is the sum of the first and corresponding elements of list, without using higher-order functions. For example:

```
addFirstA [4,3,2,1] = [8,7,6,5]
```

- (b) Repeat the problem in part a and write `addFirstB`, but you should use a higher-order function (such as `map` or `foldr`).

2. Define 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"
```

3. 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]
```

4. Write a function `deleteSecond :: (Eq a) => a -> ([a] -> [a])` that takes an item (of a type that has an `==` function defined for it) and a list, and returns a list just like the argument list, but with the second occurrence of the item (if any) removed. For example.

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

5. Write a function `associated :: (Eq a) => a -> [(a,b)] -> [b]` which takes a value of some type (call this input `x`) and a list of tuples whose first element is of `x`'s type. It should pull out all elements of list whose second tuple element are the same as `x` and return a list of these values.

For example:

```
associated 3 [(3,4), (5,7), (3,6), (9,3)] = [4, 6]
associated 2 [(1,a), (3,c), (2,b), (4,d)] = [b]
associated 'c' (zip ['c', 'c' ..] [1, 2 ..]) = [1, 2 ..]
```

6. 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).