

6.090 IAP '05 - Optional Homework 5

Assigned Friday January 13th.

Due whenever you feel like it, if you feel like it.

Submit a print-out of your .scm file to yourself or to the course staff. Should you encounter difficulty printing, the file may be emailed to yourself or the 6.090 staff.

Problem 1: List Procedures

1. Write the procedure `square-list`, which takes in a list of numbers and returns a list of their squares.

```
(square-list (list 1 2 3))  
;Value: (1 4 9)  
(square-list (list 3))  
;Value: 9  
(square-list nil)  
;Value: ()
```

2. Write the procedure `stutter-list`, which takes in a list and returns a list that contains each element of the original list twice:

```
(stutter-list (list 1 2 3))  
;Value: (1 1 2 2 3 3)  
(stutter-list (list 1))  
;Value: (1 1)  
(stutter-list nil)  
;Value: ()
```

3. Write the procedure `only-even`, which takes in a list of numbers and returns a new list containing only the even numbers from the original list.

```
(only-even (list 1 2 3 4 5))  
;Value: (2 4)  
(only-even (list 1 3 5 7 9))  
;Value: ()  
(only-even nil)  
;Value: ()
```

```
(only-even (list 2))
;Value: (2)
(only-even (list 3))
;Value: ()
```

4. Write the procedure `add-lists`, which takes in two lists of the same length and adds the elements of each of the lists together:

```
(add-lists (list 1 2 3) (list 4 5 6))
;Value: (5 7 9)
(add-lists nil nil)
;Value: ()
```

5. Write the procedure `palindrome-list`, which takes a list as input and outputs a palindromic list with those elements. A palindrome reads the same forwards as backwards. Examples: 12321, "yay", and "A man, a plan, a canal: panama" (just the letters not the spaces or punctuation). Do this to a list by doubling the length of the list and inverting the order of the second half without duplicating the last element of the list. No recursive solution needed. You may assume the list is non-empty. You may use any of the procedures we have written in class.

```
(palindrome-list (list 1 2 3))
;Value: (1 2 3 2 1)
(palindrome-list (list 1))
;Value: (1)
```

6. Write the procedure `replace-elem`, which takes in a list of numbers, a number to replace, and a value to replace it with, and returns a list with the number replaced by the value.

```
(replace-elem (list 1 2 1 2 1 2) 1 5)
;Value: (5 2 5 2 5 2)
(replace-elem (list 1 2 3 4) 3 7)
;Value: (1 2 7 4)
(replace-elem (list 1 2 3) 7 77)
;Value: (1 2 3)
```

7. Write the procedure `list-average`, which computes the average value of a list of numbers. You may use any procedures that you have written previously (two in particular will come in handy, making this a 1 liner). You may assume that the list is non-empty.

```
(list-average (list 1 2 3))
;Value: 2
(list-average (list 1))
```

```
;Value: 1  
(list-average (list 12 34 56 78 90))  
;Value: 54
```

Problem 2: Abstractions

1. Coming soon to a web page near you.