Module 1- Scheme Programming Exercise

50 Possible Points

1/29/2024

Attempt 1

In Progress
NEXT UP: Submit Assignment

Add Comment

Unlimited Attempts Allowed

Details

Scheme Programming Exercise

The purpose of this programming exercise is to learn the basic functional programming paradigm and become comfortable using recursion.

Assignment Instructions

In this homework, you are to create a number of Scheme functions. You are to follow a strict function programming style. That means you need to follow the style we used in class and use only functions, parameters, and recursion. You may write any helper functions you need, and you may use functions created for one problem to solve another. Please do not use built in Scheme functions except the ones (and variants of them) we used in class: `car`, `cdr` (and all their variants), `cons`, `null?`, `pair?`, `list?`, `number?`, append, `=`, `eq?`, `zero?`, `if`, `cond`, and all the standard arithmetic and logic functions.

Please include a comment at the top of the file giving your name, and please include a comment at the top of each function briefly explaining the function. Scheme comments start with a semicolon.

Do not nest `cond` statements. Nor have more than two `if` statements nested inside each other. Instead, rearrange your logic so that you can write your function with a single `cond` of multiple cases.

You can assume all input is in the proper format.

Write the following functions:

1. `inorder?` takes a list of numbers and returns #t if the numbers are in non-decreasing order

```
> (inorder? '())
#t
> (inorder? '(1 4 5 6 9 10))
#t
> (inorder? '(1 4 5 4 6 10))
#f
```

2. `dotproduct` takes a two vectors (lists of numbers) and computes the dot product of the vectors. If one list is longer than the other, you can ignore the extra numbers of the longer list.

```
> (dotproduct '(1 2 3) '(-2 1 5))
15
```

3. `squareroot` takes two numbers, a value and an iteration. The iteration will be an integer greater than or equal to 0. The method will compute the squareroot of the value using *iteration* rounds of Newton's method, starting with an initial value equal to the

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

```
> (squareroot 5.0 0)
5.0
> (squareroot 5.0 1)
3.0
> (squareroot 5.0 5)
2.236067977499978
> (squareroot 5 5)
2 514229/2178309
```

4. `removesubsequence` takes two lists of atoms. The first list is a *subsequence* of the second list. The method should return the second list with the first occurence of the subsequence removed. So, if the first list is `'(a b c)`, the first `a` if the second list is removed, the first `b` that appears after the removed `a` is removed, and the first `c` that appears after the removed `b` is removed.

```
> (removesubsequence '(1 3 5) '(0 1 2 3 4 5 6))
(0 2 4 6)
> (removesubsequence '(1 3 5) '(5 4 3 2 1 2 3 4 5))
(5 4 3 2 2 4)
> (removesubsequence '(a b c) '(d b c a c b a b c))
(d b c c a b)
```

5. `reverse*` takes a nested list and reverses the contents of the list and all nested lists

```
> (reverse* '(a b (c (d e ((f) g)) h)))
((h ((g (f)) e d) c) b a)
```

6. `first*` takes a list of lists and returns the first (left most) atom that appears in the list, regardless of how nested it is

```
> (first* '(a (b c) ((d e))))
a
> (first* '(((a (b c)) d) e))
a
> (first* '((() a b)))
()
```

7. `last*` takes a list of lists and returns the last (right most) atom that appears in the list, regardless of how nested it is. Give a simple solution that does not reverse the list.

```
> (last* '(a (b c) ((d e))))
e
> (last* '(((a (b c)) d) e))
e
> (last* '((() a b ())))
()
```

8. `numorder*?` takes a possibly nested list of numbers, and returns #t if the *values* of the entries in the list and all sublists are in non-decreasing order. The value of a number is the number. The value of a list is the sum of the values in that list.

```
> (numorder*? '((() ()) 1 (2) (2 3 (-1 4) 5) (((4) 5) 10) 20))
#t
> (numorder*? '((() ()) 1 (2) (2 3 (4 -1) 5) (((4) 5) 10) 20))
#f
> (numorder*? '(1 (2) (2 3 (-1 4)) 5))
#f
```

9. `vectormult` takes a row vector (a list of numbers) and matrix (a list of lists of numbers) and multiplies the vector times the matrix. The result is a vector where the ith element of the result is the dotproduct of the input vector and the ith column of the matrix. You can assume that the length of the vector matches the number of rows of the matrix.

```
> (vectormult '(1 2 -1) '((0 2 3) (1 2 0) (1 0 3)))
(1 6 0)
```

10. `matrixmultiply` takes two matrices (a list of lists of numbers) and multiplies them. You can assume the number of columns of the first matrix is equal to the number of rows of the second matrix.
in the same sublist

```
> (matrixmultiply '((1 0 1) (1 1 1) (0 1 1)) '((2 3 4) (-1 1 2) (3 1 -2)))
((5 4 2) (4 5 4) (2 2 0))
```

Submission Instructions

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

You can upload your file by adding it to the file submission area and clicking the submit assignment button. You should upload an scm and/or rkt file. If you have any questions about uploading and submitting your assignment, please reference the **Canvas Guides (https://community.canvaslms.com/t5/Student-Guide/tkb-p/student#AssignmentEnhancements)** or contact the Help Desk.

Grading

There will be a few programming exercises in the course. The purpose of the exercise is to gain experience with a specific programming language or paradigm. Each student is expected to work on the exercises on their own, but students may discuss general techniques or help debug errors. There should be no electronic copying of code.

Programming exercises account for 10% of your final grade. Please reference the **grading policies page (https://canvas.case.edu/courses/39909/pages/grading-policy-and-assessment-strategy)** for more information.

View Rubric

Programming Exercise 1

Criteria	Ratings						Pts
inorder? view longer description	**5 pts** **Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**4 pts** **Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**3 pts** **Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**2 pts** **Poor** The solution does not show an understanding of functional coding and the solution clearly will not work.	**1 pts** **Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**0 pts** **No Marks** Nothing useful to grade.	/ 5 pts
dotproduct view longer description	**5 pts** **Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**4 pts** **Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**3 pts** **Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**2 pts** **Poor** The solution does not show an understanding of functional coding and the solution clearly will not work.	**1 pts** **Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**0 pts** **No Marks** Nothing useful to grade.	/ 5 pts

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

Programming Exercise 1

Criteria	Ratings						Pts
view longer description	**Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**Poor** The solution does not show an understanding of functional coding and the solution clearly will not work.	**Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**No Marks** Nothing useful to grade.	
removesubsequence view longer description	**5 pts** **Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**4 pts** **Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**3 pts** **Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**2 pts** **Poor** The solution does not show an understanding of functional coding and the solution clearly will not work.	**1 pts** **Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**0 pts** **No Marks** Nothing useful to grade.	/ 5 pts

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

Programming Exercise 1							
Criteria	**Ratings**						**Pts**
	A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	The solution does not show an understanding of functional coding and the solution clearly will not work.	Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	Nothing useful to grade.	
first* view longer description	**5 pts** **Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**4 pts** **Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**3 pts** **Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**2 pts** **Poor** The solution does not show an understanding of functional coding and the solution clearly will not work.	**1 pts** **Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**0 pts** **No Marks** Nothing useful to grade.	/ 5 pts
last*	**5 pts**	**4 pts**	**3 pts**	**2 pts**	**1 pts**	**0 pts**	/ 5 pts

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

Programming Exercise 1

Criteria	Ratings						Pts
	written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	style solution with minor errors or unnecessary /unhelpful helper functions.	style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	show an understanding of functional coding and the solution clearly will not work.	scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	grade.	
numorder*? view longer description	**5 pts** **Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**4 pts** **Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**3 pts** **Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**2 pts** **Poor** The solution does not show an understanding of functional coding and the solution clearly will not work.	**1 pts** **Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**0 pts** **No Marks** Nothing useful to grade.	/ 5 pts
vectormult view longer description	**5 pts** **Excellent**	**4 pts** **Good**	**3 pts** **Reasonable**	**2 pts** **Poor**	**1 pts** **Minimal**	**0 pts** **No Marks**	/ 5 pts

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

Programming Exercise 1							
Criteria	**Ratings**						**Pts**
	functional style. If a helper function is used, it improves the functional style and/or readability of the code	minor errors or unnecessary /unhelpful helper functions.	mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	ng of functional coding and the solution clearly will not work.	problem. For example, a test for the empty list and the problem requires such a test.		
matrixmultiply view longer description	**5 pts** **Excellent** A correct solution written in a nice functional style. If a helper function is used, it improves the functional style and/or readability of the code	**4 pts** **Good** A good functional-style solution with minor errors or unnecessary /unhelpful helper functions.	**3 pts** **Reasonable** Either functional style with significant mistakes or mostly works but is written in an iterative style (for example, a list of functions that are done one after the other; lots of helper functions to mimic variable assignment or a loop)	**2 pts** **Poor** The solution does not show an understandi ng of functional coding and the solution clearly will not work.	**1 pts** **Minimal** Has something in scheme that works for the problem. For example, a test for the empty list and the problem requires such a test.	**0 pts** **No Marks** Nothing useful to grade.	/ 5 pts
							Total Points: 0

Choose a submission type

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)

Choose a file to upload
File permitted: SCM, RKT

or

Canvas Files

(https://canvas.case.edu/courses/39909/modules/items/1778561) (https://canvas.case.edu/courses/39909/modules/items/1778562)