Mid-Term Quiz

September 21, 2001

Matriculation No:

Instructions (please read carefully):

1. Write down your matriculation number on the question paper. DO NOT WRITE YOUR NAME ON THE QUESTION SET!

2. This is an (almost) closed-book quiz. You are allowed to bring in a cheat-sheet, i.e. an A4 sheet of paper handwritten on both sides, and which may contain any information you deem appropriate. You are, however, not allowed to use any other books, notes, sheets, memos, etc., except for the question sheet provided. The use of laptops and calculators is prohibited.

3. Each question contains an appendix with code from the text book that you may need to refer to in your answer.

4. This paper comprises 5 questions and 11 pages. The time allowed for solving this quiz is 2 hours.

5. The maximum score of this quiz is 20 marks. The weight of each question is given in square brackets beside the question number.

6. All questions must be answered correctly for the maximum score to be attained.

7. All questions must be answered in the space provided in the answer sheet; no extra sheets will be accepted as answers.

8. The back-sides of the sheets and the pages marked “scratch paper” in the question set may be used as scratch paper; you will not be allowed to use any other kind of scratch paper.

GOOD LUCK!

For examiner’s use:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Total</th>
</tr>
</thead>
</table>

1
Question 1 [ 5 marks ]

a) Write a Scheme procedure called every-second that takes in a list as its only argument and returns a list containing all the elements of even rank (i.e. every second element) from the input list. [4 marks]

Example call:

\[
\text{(every-second '(a x b y c z d))}
\]

; Value: (x y z)

Provide your answer to Qn. 1a in this box:
b) Write a non-recursive procedure (i.e. which does not call itself), that performs the same thing. (Hint: You may want to use `accumulate` to solve this question) [ 1 mark ]

Provide your answer to Qn. 1.b in this box:

Here’s the code for `accumulate`, provided for reference.

```scheme
(define (accumulate f zero l)
  (if (null? l)
      zero
    (f (car l) (accumulate f zero (cdr l)))))
```
Question 2 [ 4 marks ]

Write a Scheme procedure called sums that takes in a list of integers as its only argument and returns a list of two elements: the first is the sum of all odd-ranked numbers in the input list, whereas the second element is the sum of all even-ranked elements in the input.

Example call:

(sums '(1 2 3 4 5))
;Value: (9 6)

Provide your answer to Qn. 2 in this box:
Question 3 [4 marks]

This question asks you to build a variant of the solution to the “Towers of Hanoi” problem presented in class. We define a disk move to be a list of two numbers: the source pole and the destination pole. For example, (1 3) indicates the move of a disk from the first pole to the third.

Write a Scheme procedure called hanoi which takes in a list of 4 parameters:

- the number of disks,
- the source pole,
- the destination pole,
- the auxiliary pole,

and returns a list of disk moves that, if executed in that sequence, will move all the disks from the source pole to the destination pole and comply with the rules of the Tower of Hanoi game. (Hint: you will not get any marks for a solution that prints a sequence of moves, since that has already been given in class).

Example call:

(hanoi 3 1 2 3)
;Value: ((1 2) (1 3) (2 3) (1 2) (3 1) (3 2) (1 2))

Provide your answer to Qn. 3 in this box:
Question 4 [ 5 marks ]

Consider the following piece of code:

```
(define (unknown1 l)
  (define (unknown2 a b) (if (< a b) a b))
  (cond ((null? l) (error "Wrong parameter"))
     ((null? (cdr l)) (car l))
     (else (unknown2 (car l) (min1 (cdr l))))))

(define (unknown3 x l)
  (cond ((null? l) ()))
     ((eq? x (car l)) (cdr l))
     (else (cons (car l) (unknown3 x (cdr l))))))

(define (unknown4 l)
  (if (null? l)
     ()
     (let ((m (unknown1 l)))
       (cons m (unknown4 (unknown3 m l))))))
```

a) What is the value that the Scheme interpreter returns for the following expression? [1 mark]

```
(unknown1 (list 6 5 4 1 2 3))
```

Provide your answer to Qn. 4.a in this box:

```

```

b) What is the value that the Scheme interpreter returns for the following expression? [1 mark]

```
(unknown3 4 (list 6 5 4 1 2 3))
```

Provide your answer to Qn. 4.b in this box:

```

```
c) What is the value that the Scheme interpreter returns for the following expression? [1 mark]

\(\text{unknown4 (list 6 5 4 1 2 3)}\)

Provide your answer to Qn. 4.c in this box:

\[
\]

d) Given that the order of growth of \textit{unknown1} and \textit{unknown3} is \(O(n)\), where \(n\) is the length of the list parameter, estimate the order of growth of procedure \textit{unknown4}. [1 mark]

Provide your answer to Qn. 4.d in this box:

\[\]
Question 5 [2 marks]

Our friend Louis Reasoner has a pocket full of change. He needs to make a telephone call that will cost him \( x \) cents, and he wants to know all the ways in which he can use his change to make up that amount. Please help him in writing a Scheme procedure which takes as parameters the amount \( x \) and a list of all the coins Louis has in his pocket, and returns a list of lists, such that each sub-list of the result contains a valid combination to make up \( x \). A combination may appear more than once, since it may be using different coins of the same denomination.

So far, Louis managed to come up with the following incomplete solution:

\[
\begin{align*}
\text{(define (makeup-amount x l)} \\
\text{(cond ((pair? l) )} \\
\text{\hspace{1cm} (append (map <........> <........>)} \\
\text{\hspace{1cm} (makeup-amount x (cdr l)))))} \\
\text{((= x 0) (list () ) } \\
\text{(else () ))}
\end{align*}
\]

Please help Louis by filling in the 2 blanks \(<\ldots>\) of the above procedure.

Example call:

\[
\begin{align*}
\text{(makeup-amount 22 '(1 10 5 20 1 5 1 50))}
\end{align*}
\]

;Value: ((1 10 5 1 5) (1 10 5 5 1) (1 20 1) (1 20 1) (10 5 1 5 1) (20 1 1))

Note: The sublist (1 20 1) appears twice. Each appearance of the number 1 refers to a different coin.

For reference, here is the code for map and append:

\[
\begin{align*}
\text{(define (map f l)} \\
\text{\hspace{1cm} (if (null? l) )} \\
\text{\hspace{2cm} () )} \\
\text{\hspace{2cm} (cons (f (car l)) (map f (cdr l)))))}
\end{align*}
\]

\[
\begin{align*}
\text{(define (append l1 l2)} \\
\text{\hspace{1cm} (if (null? l1) l2 )} \\
\text{\hspace{2cm} (cons (car l1) (append (cdr l1) l2)))}
\end{align*}
\]

Please turn over for the answer box \(===>\)
Provide your answer to Qn. 4.d in this box:
Scratch Paper
Scratch Paper