End-semester Examination

Abstractions and Paradigms in Programming

Date: 25th April 2011 Weightage: 50 %
Total Marks: 100 Closed Book Exam

1. This question tests your ability to build abstractions with higher order functions. Consider the following: (18 Marks)

   (a) $tan(x)$ represented as a continued fraction. Only three terms of the fraction are shown:

   $$\cfrac{x}{1 - \cfrac{x^2}{3 - \cfrac{x^2}{5}}}$$

   (b) The nested expression of square roots

   $$\sqrt{x + \sqrt{x + \sqrt{x}}}$$

   (c) The series expansion of sin as a nested expression:

   $$\frac{x}{1!} - x^2 * (\frac{x}{3!} - x^2 * \frac{x}{5!})$$

   Note carefully that for each of the expressions, only three terms have been shown.

   Write a higher order function `hof` (whose parameters you have to decide), and express the the k-term continued fractions and nested expressions as instances of `hof`, i.e. complete the definitions shown below.

   ```
   (define (tan x  k) ... (hof ...))
   (define (nested-sqrt x k) ... (hof ...))
   (define (sin x k) ... (hof ...))
   ```

2. Suppose we have two list of numbers, and we want to convert the first list to the second by doing one of three operations. We can insert a number into the first list in any position, we can delete an occurrence of a number from the first list, or we can replace one occurrence of a number with another number. Write a function `make-equal` that computes the minimal number of changes we need to make to convert the first list to the second?

   Hint: The most interesting case is when the first elements of the two lists are not the same. (8 Marks)

3. We want to represent UNIX directories in the form of trees. For this purpose, we have defined the following structures:

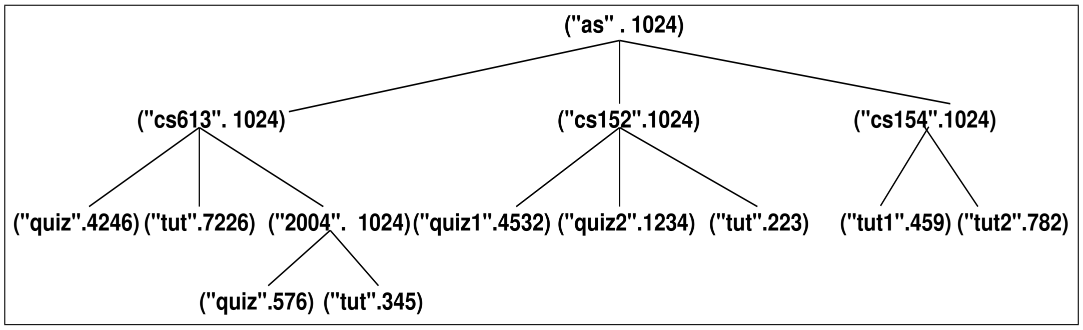
   ```
   (struct dir (info fdlist) #:transparent)
   (struct file (info contents) #:transparent)
   ```

   Here

(a) `info` is a cons-ed pair `(name.size)`, where `name` is a string representing the name of the file or the directory and `size` is an integer representing its size.

(b) `fdlist` is a list of files and directories and contents is a string representing the contents of the file.

(c) `pathname` is a list of strings. For example, `("as" "cs613" "2004")` is an example of a pathname.



(a) Write function `findtree` which takes a directory tree and a pathname and returns the subdirectory represented by the pathname.

(b) Define a function `ls` which takes a directory and a pathname and lists the names of all the files and directories in the sub-directory represented by the pathname. The files under a directory should be listed immediately after the directory is listed.

(c) Define a function called `size` which takes a directory tree and a pathname, and returns the size of the entire directory tree represented by the pathname.

(d) Define a function called delete which takes a directory tree and a pathname and returns the tree with the subdirectory or the file represented by the pathname deleted.

To keep things simple, assume that the file or the directory represented by the pathname exists in the directory tree. (20 Marks)

You can use the following example:

```
(define thistree
  (dir
   (cons "as" 1024)
   (list (dir
          (cons "cs613" 1024)
          (list (file (cons "quiz" 4246) "junk")
                (file (cons "tut" 7226) "junk")
                (dir (cons "2004" 1024)
                     (list (file (cons "quiz" 576) "junk")
                           (file (cons "tut" 345) "junk")))))
         (dir (cons "cs152" 1024)
              (list (file (cons "quiz1" 4532) "junk")
                    (file (cons "quiz2" 1234) "junk")
                    (file (cons "tut" 1223) "junk")))
         (dir (cons "cs154" 1024)
              (list (file (cons "tut1" 459) "junk")
                    (file (cons "tut2" 782) "junk"))))))
```

4. For the program shown below:

```
(define (f x)
  (define (p) (* x x))
  (define (g y)
    ((h (- y 1)) 1))
  (define (h x)
    (lambda (w) (+ w x (p))))
  (g x))
(f 2)
```

   (a) Show the execution of the program using the substitution model.

   (b) In the environmental model of execution, show the *complete environment* just when the multiplication is about to be done. (14 Marks)

5. Ashwin, Ankush, Umang, Sanket, and Krishna live in a five-storey hostel. Ashwin doesn't live on the 5th floor and Ankush doesn't live on the first. Umang doesn't live on the top or the bottom floor, and he is not on a floor adjacent to Krishna or Ankush. Sanket lives on some floor above Ankush.

   Write a predicate in Prolog called occupancy which will show which person lives in which floor. (10 Marks)

```
?- occupancy(Hostel).
Hostel = [[krishna, 1], [ankush, 2], [ashwin, 3], [umang, 4],
[sanket, 5]] ;
false.
```

6. This question is about the design of an object oriented system. You have to design a banking system consisting of four classes: An `account` class, a `joint-account` class, a `credit-card` class and a `timer` class. Their behaviour is illustrated by the following commentary. (30 Marks)

   (define this-timer (new timer%))

   There is only one `timer%` object called `this-timer`. The purpose of the timer is to trigger certain activities at designated points of time called *ticks*. For instance, on a `process-accounts-tick`. interests are paid to all accounts. Similarly, uncleared dues on a credit card are penalized on a `process-credit-card-tick`.

```
(define my-account (new account% (passwd "amitabha") (balance 1000)
(interest-rate 0.05)))

(define my-card (new credit-card% (acct my-account)(passwd "amitabha")))

(define another-account (new account% (passwd "abcd") (balance 5000)
(interest-rate 0.1)))

(define another-card (new credit-card% (acct another-account)
(passwd "abcd")))

(define jnt-account (new joint-account% (account my-account)
(passwd "amitabha") (new-passwd "xyz")))
```

   The meanings of these are obvious. Two accounts and two credit cards are created. Apart from `withdraw`, an account should also have a method called `deposit` and other methods appearing in the commentary. Each credit card is tied to an account. Each credit card has a password. A joint account is also created.

```
(send my-account withdraw 200 "amitabha")
(send my-account show "amitabha")
> 800
```

```
(send jnt-account withdraw 200 "amitabha")
>"Incorrect Passwd"
```

The joint-account cannot be operated using the original password.

```
(send jnt-account show "xyz")
>800
```

```
(send my-account pay-interest)
send: no such method: pay-interest for class: account%
```

There is a method `pay-interest` in the `account%` class. But it cannot be accessed from outside the class.

```
(send this-timer process-accounts-tick)
(send my-account show "amitabha")
>840.0
```

The time has come to pay interest of 5% on outstanding balance. Ideally the timer should generate the tick. But to keep things simple, we generate the tick from outside and inform the timer.

```
(send my-card make-purchase 100)
```

I make a purchase of 100 rupees.

```
(send this-timer process-credit-card-tick)
```

Oops. I have not cleared my credit card dues. And the `process-credit-card` tick has arrived. So I have to pay a penalty. Let me try to clear my dues quickly and see whether this works.

```
(send my-card clear-outstanding-amount)
(send my-account show "amitabha")
>720.0
```

No luck. I had to pay the penalty of 20% on my credit-card dues of 100 rupees.

```
(send my-card make-purchase 50)
(send another-card make-purchase 1000)
(send another-card clear-outstanding-amount)
(send this-timer process-credit-card-tick)
(send my-account show "amitabha")
>720.0
```

Once again I have made a purchase and did not clear my dues on time unlike the other credit-card holder. But the penalty will only show in my account when I clear my dues and my fine.

```
(send my-card clear-outstanding-amount)
(send another-account show "abcd")
>4500.0
(send my-account show "amitabha")
>660.0
```