CS 538

Midterm Exam

Monday, April 2, 2001

11:00 AM — 1:00 PM

Instructions

Answer question #1 and any three others. (If you answer more, only the first four will count.) Point values are as indicated. Please try to make your answers neat and coherent. Remember, if we can't read it, it's wrong. Partial credit will be given, so try to put something down for each question (a blank answer always gets 0 points!).

1. (1 point)
   Java is
   (a) an island in Indonesia
   (b) a slang term for coffee
   (c) a popular programming language
   (d) a small town in South Dakota

2. (a) (17 points)
   We have seen that the `map` function is very useful when we wish to apply a function to a list of values. Many variations of `map` are possible.
   Consider a function `mapB` that is called as
   `(mapB bin-fct L)`
   `bin-fct` is a binary function—a function that takes two parameters (for example, `+` or `<`). `mapB` applies `bin-fct` to each pair of adjacent values in `L`. If `L` has N values in it, `mapB` produces a list of N-1 values. Those values are the result of applying `bin-fct` to the first and second list values, then the second and third list values, etc. If `L` contains fewer than two values in it, `L` is returned (without ever using `bin-fct`).
   For example,
   `(mapB + '(1 2 3 4))` evaluates to `(3 5 7)`
   `(mapB + '(1))` evaluates to `(1)`
   `(mapB + '())` evaluates to `()`
   Write a Scheme program that implements `mapB`.

   (b) (16 points)
   Let us now consider another variant of the `map` function called `mapF`. `mapF` is called as
   `(mapF fct filter L).`
   `fct` is a function of one argument. It is applied to each element of `L` for which `filter` (a predicate) returns anything other than `#f`. Thus `filter` "filters" `L`, selecting those values to which `fct` will be applied. For example,
   `(mapF (lambda(x) (* x 2)) (lambda (y) (> y 0)) '(1 -1 0 2 -2 3 -3))`
   produces `(2 4 6)`. It does this because values ≤ 0 are filtered out, and remaining values are passed to a function that doubles its argument.
   Write a Scheme program that implements `mapF`.

3. (a) (10 points)
   Explain what `call/cc` (also known as `call-with-current-continuation`) does.

   (b) (11 points)
   `call/cc` is often used in Scheme to simulate a "throw-catch" exception mechanism as is found in Java and other languages. Explain what needs to be done in a Scheme function `f` so that it can "throw" an exception, `e`, that will be caught (and processed) by a caller of `f`. You may assume that only one caller of `f` will try to catch the exception `f` throws.

   (c) (12 points)
   Assume we have an application that accesses a medical database, `med-info-db`. Some information in this database may be **locked**. The predicate `locked?` tests if a piece of data is locked. If it is, the data must be read using the function `(unlock data password)` where `password` is known only to the owner of data.

   The following code fragment might appear in an application that accesses medical data:

```
(let ((data (lookup med-info-db info-wanted)))
    (if  (locked? data)
         (unlock data password)
        data
    )
)
```

   This code has the problem that `password` must always be provided, even if the `info-wanted` is not locked (e.g., a patient's name or phone number).
   Modify this code so that if a password is needed, a continuation function is returned to the caller. This continuation, when given the necessary password, will resume execution by unlocking the data and finishing the computation.
   You may assume that the program has available a continuation named `return` that allows you to return a value directly to the original caller (who will provide the needed password).

4. (a) (8 points)
   Explain the difference between the *scope* of an identifier and the *lifetime* of a data value.
   (b) (8 points)
   Explain the difference between *static* typing and *dynamic* typing. Give an example that illustrates the difference between static and dynamic typing.
   (c) (8 points)
   Explain the difference between overloading and polymorphism. Give an example of a programming language that supports overloading. Give an example of a programming language that allows polymorphism. What are the advantages and disadvantages of each?
   (d) (9 points)
   Explain the difference between virtual methods and ordinary methods in a language like C++ or Java. Why are virtual methods needed in these languages?

5. (a) (14 points)
   Write a Scheme function `rm-dupl` that is called as `(rm-dupl L)`. `L` is a list of atomic values (numbers or symbols). `rm-dupl` removes duplicate values in `L`, returning a list in which each value only appears once (the order of values is unimportant). For example
   `(rm-dupl '(1 2 3 4))` returns `(1 2 3 4)`
   `(rm-dupl '(1 1 1 1))` returns `(1)`
   `(rm-dupl '(a b c b c a))` returns `(b c a)`

(b) (5 points)

Extend your definition of `rm-dupl` to handle lists within `L`. That is, if the same list value appears more than once in `L`, the duplicate copies should be removed. For example

`(rm-dupl '((1) (1) 1 (2 1) (2 1) (1 2))) returns ((1) 1 (2 1) (1 2))`

(c) (14 points)

Recall that `future`, `delay` or `pcall` may be used in MultiLisp to provide for parallel evaluation or to avoid unnecessary evaluation. Use one (or more) of these features to improve the execution speed of your implementation of `rm-dupl` (either version). In what way does your use of these constructs speed execution?