Programming Languages (CS302 2007S)

Homework 3: Programming Without Recursion

Assigned: Friday, February 2, 2007
Due: Friday, February 9, 2007
No extensions!

Summary: In this assignment, you will reimplement two classic recursive algorithms without recursion.

Purposes: To give you more experience understanding recursion. To help you think about the design of languages.

Expected Time: Two to three hours.

Collaboration: You may work alone or you may work in a group of size two or three. I encourage you to work in groups on this assignment. You may discuss the assignment with anyone you'd like, provided you cite such discussions in your assignment.

Submitting: Email me your answer. More details below.

Warning: So that this exercise is a learning assignment for everyone, I may spend class time publicly critiquing your work.

Assignment

As Hoare mentions (and as you'll see in some followup readings), the addition of recursive functions to programming languages was a great advance. Of course, it is certainly possible to translate every recursive function into a corresponding iterative function. In some cases, you'll need to use a stack to simulate the recursion. In others, you will need to simulate the stack.

Part One: Exponentiation

As you may recall from CSC151, there is an elegant, logarithmic algorithm for computing x^n, for integer n. Here, I express it in Scheme.

```
(define expo
  (let ((square (lambda (val) (* val val))))
    (lambda (x n)
      (cond
        ((zero? n) 1)
        ((odd? n) (* x (expo x (- n 1))))
        (else (square (expo x (/ n 2)))))))))
```

Rewrite this algorithm without recursion. Note that your solution must still be $O(\log_2 n)$. You may not use a stack.

You can solve this problem in C, Java, or Scheme.

Part Two: Quicksort

Hoare, who seems to be providing an initial focus to this course, is also the author of Quicksort. As you likely know, the Quicksort algorithm, like most divide and conquer algorithms, is naturally recursive. It also involves a more complex type of recursion than the previous problem, in that there are two recursive calls at each level, rather than one.

Rewrite Quicksort without using recursion.

You can solve this problem in C, Java, or Scheme.

Important Evaluation Criteria

Correct, working, solutions will earn a check.

Particularly elegant solutions, or solutions that include an analysis of the principles you used to rewrite the algorithms are likely to receive a higher grade.

Non-working or incorrect solutions will earn a lower grade.

Submitting Your Homework

Please submit this via email, using a title of CSC302 HW3.

Attach the files that you've created to that email message.