Homework Six Solutions

Due Friday, November 1st, 2002, at 5:00pm.

Your task

1. An imperative program

The following program fragment is written in the imperative programming language C. Describe (in English) what changes might occur in the machine code this program compiles to if the line `int x;` is replaced by `float x;`.

```
int x;
x = 2;
x = x+3;
```

Solution:

Changing the declaration of `x` might mean that (a) `x` is stored in a different amount of space, causing the compiler to reserve more room for it in the compiled program; (b) `x` is stored in floating-point format instead of two's-complement format, meaning that the assignment `x=2;` would copy in a different pattern of bits; (c) the addition operation in the last line would be a floating-point addition instead of an integer addition, probably involving calling a different machine-code operation.

2. A functional program

Below is a definition of a function written in the functional programming language Scheme. (See the notes linked to from the lecture schedule if you need to brush up on Scheme, but be aware that these notes use the word `function` where most dialects of Scheme use `lambda`.)

```
(define improve
    (lambda (f x)
        (if (> (f x) x)
            (f x)
            x)))
```

1. What is the result of calling this function as `(improve (lambda (x) (+ x 1)) 5)`? *Solution: 6*

2. What is the result of calling this function as `(improve (lambda (x) (- x 1)) 5)`? *Solution: 5*

3. Give a succinct description (in English) of what `improve` does. *Solution: The functon `improve` calls a function on a value and returns the larger of the result of the function call and the original value.*

3. Choosing a paradigm

Suppose you want to write a program that contains representations of bank accounts that support operations like withdrawals, deposits, balance queries, and so forth, but you want to guarantee that under no circumstances will the balance in any bank account ever go below zero (e.g., by making withdrawals fail if they are too big). What programming paradigm would make this task easiest, and why?

Solution:

This is a job for object-oriented programming, because in an object-oriented programming language we could hide the value of the bank account inside an object that could only be accessed by methods named `deposit`, `withdraw`, `balance`, etc. Enforcement of the minimum balance guarantee is then just a matter of writing some code in `withdraw`; since nothing in the program can reduce the balance without calling `withdraw`, there is no way to violate the guarantee.

In contrast, a program written in an imperative language would have to expose the value of the bank account (allowing someone to change it just by writing `bank_account_value = -50;`). A program in a functional language would either have to expose the value of the bank account (if `deposit`, `withdraw`, etc.) were separate functions, or solve the problem by burying the value in a function that supported all of `withdraw`, `deposit`, etc. In the latter case it would be easier to solve the problem in a language that provided built-in objects instead of making us reinvent them.

Submitting your solution

Write up an email message containing your full name and the answer to these problems in plain text format (this means no HTML or Microsoft word documents), and send it to `aspnes+110-02-6@cs.yale.edu`. (Note: this is **not** the same email address as for previous homework.)