CSCI-GA-2110 – Problem Set 3 – Written Part

This document describes the written exercises for problem set 3. Each exercise is designated as a "Task" in this document. Please write or type your solutions neatly to these tasks, produce a legible PDF clearly indicating where each question is answered, and upload the results to gradescope.

1 Bad If Statements

In the programming assignment, you are asked to implement an interpreter for a language with if statements. Here is an **incorrect** implementation of 'ifC' statements for that language using a store-passing style like we saw in class:

```
(define (eval-env (env : Env) (sto : Store) (e : Expr)) : Result
  (type-case Expr e
             ...
             [ifC (guard e1 e2)
                  (let* ([rguard (eval-env env sto guard)]
                         [re1 (eval-env env (res-s rguard) e1)]
                         [re2 (eval-env env (res-s rguard) e2)])
                        (cond
                          [(equal? (res-v rguard) (boolV #true)) re1]
                          [(equal? (res-v rguard) (boolV #false)) re2]
                          [else (error 'eval-env "ifC guard was not a boolean")]))]
             ...))
```

where `env` and `sto` and `Result` are as in `lecture6c.rkt` from class.

Task 1.1 (4 pts). Explain what is wrong with this implementation of `ifC`. In particular, give an example of a small program that leads to an error during evaluation using this style of `ifC`, but which would not cause an error under a correct interpretation.

2 Boxes

Consider the following Racket function:

```
(define (myfun b1 b2)
  (begin
    (set-box! b1 1)
    (set-box! b2 2)
    (+ (unbox b1) (unbox b2))))
```

If we execute `(myfun (box 0) (box 0))` it returns the value `3`.

Task 2.1 (3 pts). Give an example of how, on certain inputs, evaluting `myfun` causes it to return `4` instead.

3 Dynamic Scope and Recursion

In class we discussed how in a Racket example like the following

```
(let ([fact (lambda (n) (if (equal? 0 n) 1 (* n (fact (+ n -1)))))])
     (fact 5))
```

we get an unbound identifier error for the recursive occurrence of `fact` in the definition of `fact`. We saw how this happens by considering the behavior of this example with a substitution-based interpreter with our environment-based intepreter that had lexical scope.

Task 3.1 (3 pts). What happens when this is example is executed with dynamic scope instead? Explain why.