Homework 3

CSC 4101, Fall 2015

Due: 27 October 2015

1. (10 pts)
When the eq? function in Scheme is used for comparing symbols it must return #t if and only if the the two symbols have the same name (where uppercase and lowercase characters are not distinguished). E.g.,

(eq? 'x 'x) returns #t
(eq? 'x 'X) returns #t
(eq? 'x 'y) returns #f

With our current implementation, the parser constructs a new Ident object for every occurrence of a symbol. For comparing symbols, it is, therefore, necessary to use string comparison for comparing the names stored in these Ident objects. Explain, how you would modify your processing of symbols (data structure, lexical analysis, and parsing), so that eq? could be implemented using simple pointer comparison, i.e., using == in C#, C++, or Java.

2. (10 pts)
Suppose, we need to debug somebody else’s program. We suspect that there is a problem with the method wizbang() in class Widget or with how that method is called. We cannot modify class Widget, nor can we modify the client code that contains the calls to Widget.wizbang(), since we don’t have those sources. However, we can modify the code where Widget objects are created and we can create new classes.

In order to better understand what this method does, we would like to print the values of the parameters and the return value into a log file every time Widget.wizbang() is called. Explain, how you would produce this log file given the constraints that neither class Widget nor the client can be modified.

3. (10 pts)
Given the following Java code:

```java
class C {
    public int foo(C p) { return 1; }
}

class D extends C {
    public int foo(C p) { return 2; }
    public int foo(D p) { return 3; }
}
```

C p = new C();
C q = new D();
D r = new D();

int i = p.foo(r);
int j = q.foo(q);
int k = q.foo(r);

(Remember that in Java every object is accessed through a pointer and that methods are virtual by default.)

Which methods get called in the three calls, i.e., what are the values of i, j, and k? Explain how the method selection works to result in these values.

4. (10 pts)
Explain what the following Scheme code is doing:

```
(define (make-stream n f)
  (define (next m)
    (cons m (lambda () (next (f m)))))
  (next n))
(define head car)
(define (tail stream)
  ((cdr stream)))
(define (nth stream n)
  (if (= n 0) (head stream)
          (nth (tail stream) (- n 1))))
(define even (make-stream 0 (lambda (n) (+ n 2))))
```

Try it out in Scheme48 and check the values of the following expressions:

```
even
(head even)
(head (tail even))
(head (tail (tail even)))
(head (tail (tail (tail even))))
(nth even 5)
(nth even 1000)
```

Explain what the lambda in make-stream is good for, where this function is called, and how tail and nth work. To see what’s going on, trace manually through the execution of

```
(head (tail (tail even)))
```