EECS 490 Midterm Exam, Fall 2016

Time: 75 minutes

- This exam is closed book, closed notebook.
- You may not provide your own scratch paper, but the last two pages are intentionally left blank for scratch work.
- Laptops, cell phones, calculators, and smartwatches are not allowed.
- Cell phones must be turned off.
- You are allowed one 8.5x11" sheet of notes as a reference.
- Any deviation from these rules will constitute an Honor Code violation.

- The exam consists of 5 questions, each with multiple parts.
- For Question 5, choose one of the two options to answer; we will only grade one answer.
- For multiple-choice and true/false questions, indicate your choice clearly.
- For short-answer and coding questions, write your answers clearly in the space provided.
- Do **not** write in the margins of a page (i.e. within an inch of the border), as anything written there will not show up when your exam is scanned.
- Assume all Python code is Python 3.5, all C++ code is C++11, and all Scheme code is R5RS. Adhere to these standards in the code you write as well.
- Your exam should have 10 pages, including this page and the two blank pages at the end.

Include your signature to indicate that you acknowledge the Honor Pledge, printed below.

I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.

Name ____

Uniqname ____

UMID ____

Signature ____

Name of person to your left ________________________________

Name of person to your right ________________________________

Question 1 (20 points)

Select the correct answer for each problem below from choices A-E. There is no penalty for an incorrect answer, and no credit will be awarded for a problem left blank or with multiple choices selected.

a) Given that `a` is an integer and `p` is a pointer to an integer, which of the following C++ expressions produce an l-value:

```
  I) a + *p
 II) a--
III) ++a
 IV) *p
  V) a = *p
```

A. II and IV

B. IV and V

C. III, IV, and V

D. II, IV, and V

E. I, III, IV, and V

b) Consider the following code in a C-like language that uses **dynamic scope**. What does it print?

```
int a = 1, b = 2, x = 3, y = 4;

void f1() {
  print(a + x);
}

void f2(int x, int y) {
  f1();
}

void f3(int a, int b) {
  f2(5, 6);
}

int main() {
  f3(b, a);
}
```

A. 4

B. 5

C. 6

D. 7

E. This code will result in a compile-time or runtime error

c) Which of the following regular expressions match the entire string `"abbc"`:

```
  I) [abc]
 II) (a|b|c)*
III) a+b+c
 IV) [a-c]*
  V) a|b|c*
```

A. III

B. II and IV

C. III and V

D. I, III, and V

E. II, III, and IV

d) Which of these fragments of Scheme code introduce a new frame:

```
  I)  (let ((x 3)) (+ x 3))
 II)  (begin x 3)
III)  (define x 3)
 IV)  (define (x) 3)
  V)  (lambda (x) 3)
```

A. I

B. I and II

C. I and V

D. I, IV, and V

E. I, III, IV, and V

Question 2 (24 points)

For each statement below, circle **True** or **False**, or leave it blank. Unanswered statements will receive 50% credit.

a) **True** / **False** In most languages that provide exceptions, exception handlers are dynamically scoped.

b) **True** / **False** In lambda calculus, evaluation of an expression terminates when no more beta-reductions can be applied.

c) **True** / **False** In languages that allow both mutation and lambda functions, it is always the case that the code within a lambda function is able to modify variables from the non-local environment.

d) **True** / **False** In Scheme, the `cdr` of a pair must be either another pair or the empty list (i.e. `nil`).

e) **True** / **False** Python uses the call-by-reference convention for parameter passing.

f) **True** / **False** In C++, a function that uses the `goto` statement can always be rewritten to provide the same behavior without a `goto`.

g) **True** / **False** In C++, types are first-class entities.

h) **True** / **False** Any algorithm that can be written in C++ can also be written in Scheme.

i) **True** / **False** In a statically typed language such as C++, the programmer must declare the type of every expression in a program.

j) **True** / **False** As long as a binding for `x` is in scope, the following Scheme expression will never result in a runtime error:

```
(if (pair? x) (cdr x) '())
```

k) **True** / **False** In C++, any code that is legal according to the grammar of C++ will not result in a compile-time error when it is compiled.

l) **True** / **False** Invoking a continuation causes the state of data to be restored to what it was at the time the continuation was created.

Question 3 (26 points)

a) Translate the following Scheme expression into an equivalent expression that uses a single `let` form instead of the `lambda`:

```
((lambda (x y)
   (if (> x y)
       (- x 1)
       y))
 3 1)
```

b) Evaluate the λ-calculus term below until it is in normal form, using the normal-order evaluation strategy. Show each α-reduction and β-reduction step, as in the following:

$$\begin{array}{lll} & (\lambda x.\, x)\ (\lambda x.\, x) & \\ \rightarrow & (\lambda x.\, x)\ (\lambda y.\, y) & (\alpha\text{-reduction}) \\ \rightarrow & \lambda y.\, y & (\beta\text{-reduction}) \end{array}$$

Term to evaluate:

$$(\lambda x.\ \lambda y.\ x\ y\ \lambda z.\ z)\ (\lambda u.\ \lambda w.\ u\ w)$$

c) Consider the following context-free grammar, with start symbol S:

$$S \rightarrow a\ S\ b \mid C$$
$$C \rightarrow C\ c \mid \varepsilon$$

i. Draw the derivation tree for the string: *aaccbb*

ii. Describe in one sentence the set of strings that are matched by the grammar.

d) In no more than three sentences, explain the distinction between frames and environments.

Question 4 (15 points)

The *sieve of Eratosthenes* is a method for computing prime numbers. Given a set that initially contains all the natural numbers starting from 2, the algorithm is as follows:

1. Let *k* be the smallest number still in the set. It must be the case that *k* is prime.
2. Eliminate all multiples of *k* from the set.
3. Go to step 1.

In this problem, implement the sieve of Eratosthenes as a Python generator. We will use the following generator for the natural numbers:

```
def naturals():
    crnt = 0
    while True:
        yield crnt
        crnt += 1
```

a) First, implement the `make_is_not_multiple()` function below, that given a number `k`, returns a function that produces `False` when applied to a multiple of `k` and `True` otherwise.

```
def make_is_not_multiple(k):
    """Returns a function that tests whether a number is not a
    multiple of k.

    >>> fn = make_is_not_multiple(5)
    >>> fn(3)
    True
    >>> fn(10)
    False
    """
    # your code below
```

b) Now complete the definition of the `primes()` generator below. You may also use the built-in Python `filter()`, whose documentation is as follows:

```
>>> help(filter)
Help on class filter in module builtins:

class filter(object)
|  filter(function or None, iterable) --> filter object
|
| Return an iterator yielding those items of iterable for which
| function(item) is true. If function is None, return the items
| that are true.
```

Fill in your code below:

```
def primes():
    """A generator that produces the prime numbers in order.

    >>> p = primes()
    >>> [next(p) for i in range(10)]
    [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
    """
    numbers = naturals()
    next(numbers)  # discard 0
    next(numbers)  # discard 1
    # your code below
```

Question 5 (15 points)

For this question, choose one of the problems below to implement. We will only grade your solution to one of the questions. If you write code for both, you must clearly indicate which one you want us to grade, or we will choose one arbitrarily. You must write your code in R5RS Scheme using the standard forms that were in Project 1. You may use the following primitive procedures: `null?`, `pair?`, `list?`, `eq?`, `not`, `cons`, `car`, `cdr`, `list`, `append`. Your functions do **not** have to be tail recursive. You may write any helper functions you want.

a) Write a function `deep-map` that maps a function across integers stored in an arbitrary nesting of pairs. The return value should maintain the same pair structure, but contain values that are the result of applying the function to each value in the original structure. The following are examples of using this function:

```
> (deep-map (lambda (x) (+ x 1)) 3)
4
> (deep-map (lambda (x) (+ x 1)) '())
()
> (deep-map (lambda (x) (+ x 1)) '(1 (2 3) (4 . 5) (6 (7 8))))
(2 (3 4) (5 . 6) (7 (8 9)))
```

b) Write a function `unique` that takes in a list of items and produces a new list that contains the items from the original list but without duplicates. The resulting list is allowed to be in any order. The following are examples of using this function:

```
> (unique '())
()
> (unique '(1))
(1)
> (unique '(1 2 3))       ; any ordering in the result is fine
(1 2 3)
> (unique '(1 2 1 3 2))   ; any ordering in the result is fine
(3 1 2)
```

Write your code below

This page intentionally left blank. We will not grade any work here.

This page intentionally left blank. We will not grade any work here.