# EECS 490 Midterm Review Exercises

## True/False

F 1. Any programming language that supports the object-oriented paradigm also supports functions as first-class entities.

T 2. In C++, the `goto` statement can be used to accomplish the same effects as `break` and `continue`.

T 3. Exceptions enable control to be transferred from a function to another one that is not the direct caller of the function.

T 4. In Python, both a `try` with a `finally` clause and a `with` statement can be used to ensure that a piece of code runs whether or not an exception is raised.

F 5. In C++, the language standard requires that objects that are no longer reachable by a running program be automatically reclaimed by the runtime.

F 6. In both Python and C++, the body of a loop has a frame associated with it that contains variables that are only accessible from within the loop. yes in C++, no in Python

F 7. In a language with static scope, reference semantics, and call by value, it is possible to write a function that swaps the values of two variables in the caller's local environment.

T 8. In C++, a context manager can be implemented by placing code for acquiring and releasing a resource in the constructor and destructor of an object.

F 9. In languages that provide static storage duration, an object with static storage duration must be initialized when the program starts. can be initialized on first use

F 10. A language that supports default arguments must also provide keyword arguments. see C++

T 11. Variadic arguments in Python are placed in a tuple for standard arguments and a dictionary for keyword arguments.

T 12. In call by name, an argument expression is not evaluated before the body of the invoked function starts executing.

F 13. In Scheme, functions can be defined using any of the special forms `define`, `lambda`, or `let`.

F 14. In Scheme, the argument to a `quote` form is evaluated by the interpreter.

T 15. Any recursive algorithm can be rewritten to use iteration instead. but may require explicit storage

T 16. In Python, decorating a function definition causes the decorator to be invoked on the result of executing the function definition.

F 17. In C++, a lambda expression causes the environment of the enclosing function to be maintained until the lambda function is no longer in use.

T 18. The Java compiler infers the type of a lambda object based on the context in which it is used.

T 19. In both Python and C++, it is possible to specify a subset of a function's arguments at an earlier time then the rest of the arguments. functools.partial() in Python, std::bind in C++

T 20. First-class continuations can be used to implement a mechanism for exceptions.

T 21. Any set of strings that can be expressed with a regular expression can also be expressed with a context-free grammar.

F 22. Evaluation of an expression in lambda calculus always terminates. $(\lambda x. x\ x)(\lambda x. x\ x)$

## Free Response

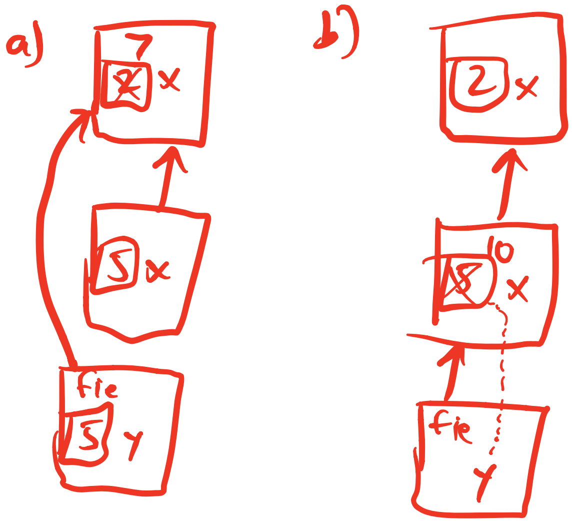
1. (4.6-7 in Gabbrielli and Martini) Consider the following pseudocode:

```
{
  int x = 2;

  int fie(int y) {
    x = x + y;
  }

  {
    int x = 5;
    fie(x);
    print(x);
  }

  print(x);
}
```

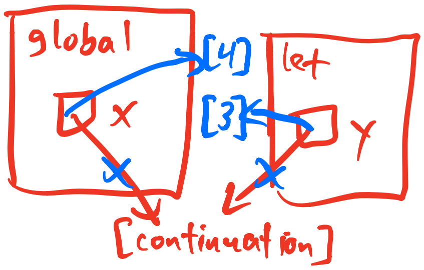


a) Determine what is printed by the code if the language uses static scope and call by value. 5 7

b) Determine what is printed if the language uses dynamic scope and call by reference. 10 2

2. What is the value of x after the following Scheme code is evaluated? 4

```
> (define (foo c) c)
> (define x
    (let ((y (call/cc foo)))
      (if (number? y)
          (+ y 1)
          y)))
> (x 3)
```



3. Write a Scheme function `remove` that takes an item and a list and produces a new list with all occurrences of that item removed. Some examples:

```
> (remove 3 '(1 2 3 4))
(1 2 4)
> (remove 3 '(1 2 4))
(1 2 4)
> (remove 3 '(3 1 3 2))
(1 2)
```

```
(define (remove item lst)
  (if (null? lst) lst
    (let ((x (car lst))
          (rest (remove item (cdr lst))))
      (if (eq? item x) rest (cons x rest)))))
```

4. (1.43 in SICP) Write a Scheme function `repeated` that takes a function `fn` and a number `n` and returns a new function that applies the `fn` a total of `n` times on its argument. Example:

```
> ((repeated (lambda (x) (+ x 1)) 5) 3)
8
> ((repeated (lambda (x) (* x x)) 3) 2)
256
```

```
(define (repeated fn n)
  (if (= n 0) (lambda (x) x)
    (lambda (x) (fn ((repeated fn
                       (- n 1))
                     x)))))
```

5. Write a regular expression that matches strings containing zero or more occurrences of the substring "ab", followed by one or more occurrences of "c", followed by an optional "d". Examples of strings that should be matched: c, cd, abc, abcd, abcc, abccd, ababcc, ababcd.

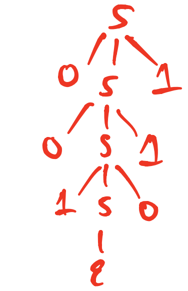
(ab)*c+d?

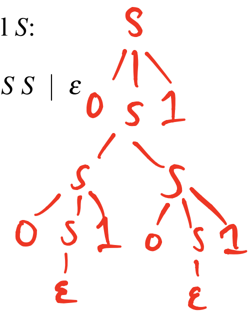
6. Describe the set of strings matched by the regular expression `(a[b-d])*(e|f)+`.

Zero or more occurrences of ab, ac, and ad, followed by one or more occurrences of e and f.

7. Given the following context-free grammar, with start symbol $S$:

$$S \rightarrow 0\,S\,1 \mid 1\,S\,0 \mid S\,S \mid \varepsilon$$





a) Draw the derivation tree for the string "001011".

there are infinitely many

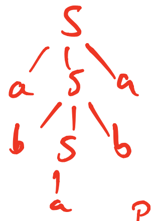
here are two

Binary strings with equal number of zeros and ones.

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

8. Given the following context-free grammar, with start symbol $S$:

$$S \rightarrow a\,S\,a \mid b\,S\,b \mid a \mid b \mid \varepsilon$$



a) Draw the derivation tree for the string `"ababa"`.

b) Describe in one sentence the set of strings that are matched by the grammar. Palindromes of a's and b's.

9. Evaluate the $\lambda$-calculus term below. Show each $\alpha$-reduction or $\beta$-reduction step, as in

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

Term to evaluate:

$$(\lambda x.\, \lambda y.\, \lambda x.\, y\, x)\ ((\lambda z.\, z\, z)\ (\lambda z.\, z\, z))\ (\lambda x.\, x)\ (\lambda y.\, y\, y)$$

More $\lambda$-calculus exercises can be found here with solutions, courtesy of UMD.

10. Define a function in $\lambda$-calculus that computes one number raised to the power of another. You may use the *times* function shown in lecture.

$$\mathit{power} = \lambda b.\, \lambda e.\, [\text{fill in the body}]$$ $e\ (\mathit{times}\ b)\ 1$

-or- $e\ b$

$$\begin{aligned} \rightarrow\ & (\lambda y.\, \lambda x.\, y\, x)\ (\lambda x.\, x)\ (\lambda y.\, y\, y) & (\beta) \\ \rightarrow\ & (\lambda y.\, \lambda x.\, y\, x)\ (\lambda z.\, z)\ (\lambda y.\, y\, y) & (\alpha) \\ \rightarrow\ & (\lambda x.\, (\lambda z.\, z)\, x)\ (\lambda y.\, y\, y) & (\beta) \\ \rightarrow\ & (\lambda x.\, x)\ (\lambda y.\, y\, y) & (\beta) \\ \rightarrow\ & \lambda y.\, y\, y & (\beta) \end{aligned}$$