Programming Languages: Final Exam

Code: 87858

1. An iterative process is generally characterized by executing in:

   (A) non-constant space
   (B) non-constant time
   (C) constant time
   (D) constant space

2. A Scheme function executes in constant space under what conditions? Choose the most general answer.

   (A) the number of non-tail recursive calls is bounded by a constant
   (B) the number of non-tail recursive calls is zero
   (C) the number of non-tail recursive calls is unbounded
   (D) the number of tail recursive calls is bounded by a constant
   (E) the number of tail recursive calls is zero
   (F) the number of tail recursive calls is unbounded

3. Which functions implement iterative processes?

```
(define (sum n total)
    (cond
        ((= n 0)
            total)
        ((< n 3)
            (+ 1 (sum (- n 1) total)))
        (else
            (sum (- n 1) (+ total n)))
        )
    )
```

```
(define (pow b e total)
    (cond
        ((= e 0)
            total)
        ((= (% e 2) 1)
            (* b (pow b (- e 1) total)))
        (else
            (pow (* b b) (/ e 2) total))
        )
    )
```

   (A) *sum*
   (B) *pow*
   (C) *sum* and *pow*
   (D) neither *sum* nor *pow*

4. Consider this example of a rewrite of a non-tail recursive function (left) to a tail recursive form (right):

```
(define (f x)
    (if (= x 0)
        1
        (* x (f (- x 1)))
        )
    )
```

```
(define (f x)
    (define (iter a b)
        (if (= b 0)
            a
            (iter (* a b) (- b 1))
            )
        )
    (iter 1 x)
    )
```

   Note the original function has one formal parameter, which is updated in the recursive call. The original function also has a base case return value of 1. Suppose you wish to do a similar rewrite of a non-tail recursive function with six formal parameters and three base cases. Two of the formal parameters are updated in the recursive call. How many arguments will the iterative helper function have in this typical tail-recursive rewrite?

   (A) five
   (B) four
   (C) two
   (D) three
   (E) seven
   (F) six
   (G) nine
   (H) eight

5. Continuing with the previous question, let A represent the set of formal parameters that are updated and B represent the set of formal parameters that are not updated in the recursive call. What is sent to the iterator in the initial call?

   (A) the base case return values
   (B) the initial values of A and B
   (C) the base case return values and the initial values of A and B
   (D) the base case return values and the initial values of B
   (E) the initial values of B
   (F) the initial values of A
   (G) the base case return values and the initial values of A

6. Assuming the functions referenced below have the semantics implied by their names, what value does the following code produce?

```
(accumulate
    +
     0
     (map
         (lambda (z) (- z 1))
         (filter-keep-if-predicate-true
             (lambda (n) (or (= (remainder n 2) 0) (= (remainder n 3) 0)))
             (enumerate-integers
                 1  ;from this, including
                 18 ;to this, including
                 )
             )
         )
     )
```

(A) 88
(B) 105
(C) 33
(D) none of the listed answers are correct
(E) 117
(F) 36
(G) 97
(H) 17

For the next set of questions, draw an environment picture in the style of the text after the following code is evaluated. Label successive tables *E1*, *E2*, *E3*, ... as they come into existence, with the global table labeled E0.

```
(define w 3)
(define (g x) (if (= x 0) 0 (g (- x 1))))
(define (f x)
    (define (g y)
        (lambda (z)
            (+ w x y z)
            )
        )
    g
    )
(define a ((lambda (x) (+ x w)) w))
(define b (f a))
(define c (b 5))
(define d (g 1))
; mark
```

7. How many tables are created, not including the global table?

(A) seven
(B) five
(C) four
(D) one
(E) six
(F) eight
(G) two
(H) three

8. Which tables may be garbage collected at the `; mark` comment?

(A) E1, E4, E5
(B) E2, E3, E4
(C) E1, E2, E3, E6
(D) E2, E3
(E) E1, E2, E3, E6
(F) E6, E7
(G) none of the listed answers are correct
(H) E6, E7, E8

9. Which variables in E0 are bound to integers?

(A) w, c, d
(B) w, a, c, d
(C) w, d
(D) none of the listed answers are correct
(E) w, a, d
(F) w, d
(G) w, a, c
(H) w, c

10. How many closures are created?

(A) three under E0, one each under E2 and E3
(B) two under E0, one under E3, two under E5
(C) three under E0, one each under E1, E2, and E3
(D) two under E0, one each under E2 and E3
(E) none of the listed answers are correct
(F) two under E0, two under E2, two under E3
(G) two under E0, one each under E3, E4, and E5
(H) three under E0, one under E2

11. How many closures may be garbage collected at the `; mark` comment?

(A) one
(B) none
(C) five
(D) two
(E) four
(F) three

12. How many tables directly point to E0?

(A) one
(B) four
(C) six
(D) none
(E) seven
(F) two
(G) five
(H) three

13. What are the first five elements of the stream s:

```
(define (make-mystery-stream factor base)
    (cons-stream
        (* factor base)
        (make-mystery-stream
            (* factor base -1)
            base
            )
        )
    )
(define s (make-mystery-stream 1 2))
```

(A) 2 -4 6 -8 10
(B) 2 2 -2 -2 2
(C) 2 2 2 2 2
(D) 2 4 8 16 32
(E) 2 -2 2 -2 2
(F) 2 -4 8 -16 32
(G) none of the listed answers are correct
(H) 2 4 6 8 10

14. What are the first six values of this stream¿

```
(define s (cons-stream 0 (cons-stream 1 (add-streams s s))))
```

(A) 1 1 2 2 3 3
(B) 0 1 1 2 3 5
(C) none of the listed answers are correct
(D) 1 1 2 3 5 8
(E) 0 1 0 2 0 4
(F) 1 2 4 8 10 12
(G) 1 2 6 24 120 720
(H) 1 0 2 0 6 0

15. Consider this function definition and call:

```
(define (f x) (cons x (lambda () (f (lambda () (+ (x) 1))))))
(f (lambda () 1))
```

Suppose all calls to *cons* wrap the second argument in a lambda, as shown above. How would *car* and *cdr* need to be modified, respectively, to yield stream-like behavior?

(A) modify both *car* and *cdr*, calling the lambdas
(B) modify *cdr* only, calling the lambda
(C) it is not possible to get stream-like behavior
(D) no changes are necessary
(E) modify *car* only, calling the lambda

16. **T** or **F**: Java implements *call-by-value* for simple variables (i.e. variables holding values such as integers or reals), but implements *call-by-reference* for object variables.

17. **T** or **F**: Without using an alias, it is still possible to distinguish *call-by-reference* from *call-by-value-result* in a single-threaded system.

18. What is a difference between a *non-hygenic* macro and a *call-by-name* function call?

(A) a macro renames locals while *call-by-name* does not
(B) a macro may evaluate arguments more than once; not so for *call-by-name*
(C) *call-by-name* renames locals while a macro does not
(D) *call-by-name* may evaluate arguments more than once; not so for macros
(E) they are the same

19. Given a constant k, this loop:

```
while (z > 1)
    {
    x = x + 2;
    z = z - 1;
    }
```

and the invariant $z + x/2 = k$, proving `<<requires: I AND E>> S <<ensures: I>>` leads to which of the following equations? Choose the simplest answer.

(A) $z + x/2 = k \wedge z \geq 1 \rightarrow (z-1) + (x+2)/2 = k$
(B) $z + x/2 = k \wedge z \geq 1 \rightarrow z + x/2 = k$
(C) none of the listed answers are correct
(D) $z + x/2 = k \wedge z > 1 \rightarrow (z-1) + (x+2)/2 = k$
(E) $(z+1) + (x+2)/2 = k \rightarrow z + x/2 = k$
(F) $z + x/2 = k \wedge z < 1 \rightarrow z + x/2 = k$
(G) $(z+1) + (x+2)/2 = k \wedge (z+1) > 1 \rightarrow z + x/2 = k$
(H) $z + x/2 = k \wedge z \leq 1 \rightarrow (z-1) + (x+2)/2 = k$

20. What is the weakest precondition for this loop and postcondition?

```
while (z > 1)
    {
    x = x + 2;
    z = z - 1;
    }
<<x = z>>
```

Choose the simplest answer.

(A) $(x + 2 * i = z - i) \vee (x = z \wedge z <= 1)$
(B) none of the listed answers are correct
(C) $x + 2 * z = 3 \vee (x = z \wedge z <= 1)$
(D) $x + 2 * z = 0$
(E) $x + 2 * z = 0 \vee (x = z \wedge z <= 1)$
(F) $x + 2 * i = z - i) \wedge z = i + 1$
(G) $(x + 2 * i = z - i \wedge z = i + 1) \vee (x = z \wedge z <= 1)$
(H) $x + 2 * z = 3$