

91.301, Organization of Programming Languages  
Fall 2005, Prof. Yanco

### Sample Quiz 1

#### Problem 1 (9 points)

What is the result of the evaluation of the final statement in each of the following groups of Scheme expressions? Write “error” if an error would result or “procedure” if a procedure would be returned. Assume that each group is evaluated separately.

```
(define a 2)
(define b 4)
(define c 6)

(let ((a 4)
      (b (+ a 1))
      (c b))
  (+ a b c))
```

```
(define (square x) (* x x))

(define (my-proc a) (lambda (x) (a (a x))))

(my-proc square)
```

```
(define (c f)
  (lambda (x) (f (f (f (f x))))))

(define (dec x)
  (- x 1))

((c dec) 8)
```

**Problem 2 (9 points)**

Assume the following Scheme expressions are evaluated sequentially. Fill in the table below, using `eq?`, `eqv?` and `equal?` to compare the items in the first two columns. You may find it useful to draw the box and pointer diagrams for this problem, but you are not required to do so.

```
(define a (list 1.2 (list 2 3) 4))
```

```
(define b (cons 1.2 a))
```

```
(define c (cons 1.2 a))
```

		eq?	eqv?	equal?
b	c			
(car a)	(car b)			
(cdr b)	(cdr c)			

**Problem 3 (8 points)**

For the two expressions below, write the sequence of cars and cdrs needed to get the number 3 out of each list. You may find it helpful to draw the box-and-pointer diagrams, but you will not be graded on them.

```
(define first-list (list (list 1 2 3) 4 5))
```

```
(define second-list (cons 1 (cons (list 2) (list 3 4))))
```

**Problem 4 (10 points)**

Given that `inc` adds one to its argument

`(inc x) → x+1`

and `dec` subtracts one from its argument

`(dec x) → x-1`

what does `fun` do?

```
(define (fun a b)
  (if (= a 0)
      b
      (inc (fun (dec a) b))))
```

What is the order of growth of the `fun` procedure in terms of time and space?

Time: \_\_\_\_\_

Space: \_\_\_\_\_

What variable in the function is  $n$  dependent upon?

Does `fun` generate a recursive or an iterative process? \_\_\_\_\_

**Problem 5 (10 points)**

Write a function `apply-backwards` that takes a function `f` as its argument and returns a function that takes two arguments as input and returns the value that one would obtain if `f` were applied to those arguments in reverse order.

For example,

```
((apply-backwards -) 4 5)
```

would return

```
1
```

and

```
((apply-backwards /) 3 6)
```

would return

```
2
```

**Problem 6 (16 points)**

The `prod` function defined below computes the product from  $i=a$  to  $b$  of some function  $f(i)$ , incrementing  $i$  as defined by the `next` function. (Note that this is quite similar to the `sum` function that we wrote in class.)

```
(define (prod term a next b)
  (if (> a b)
      1
      (* (term a)
         (prod term (next a) next b))))
```

a. Use the `prod` function to compute the product of every third number between  $a$  and  $b$  (that is,  $a, a+3, a+6$ , etc.).

```
(define (prod-of-every-third a b)
  (prod _____
         _____
         _____
         _____))
```

b. Define a function called `fact` to compute the factorial of  $n$  using the `prod` function defined above. For example, `(fact 5)` should return 120.

**Problem 7 (18 points)**

a) Write a procedure called `combine-two-lists` which takes 3 arguments: a procedure and two lists of numbers. The procedure should return a list whose elements are the result of applying the procedure to the corresponding elements of the original lists. For example,

```
(combine-two-lists * '(1 20 33) '(10 2 3))
```

returns the list

```
(10 40 99)
```

You may assume that the lengths of the two lists are equal.

What is the order of growth of your procedure in terms of time and space?

Time: \_\_\_\_\_

Space: \_\_\_\_\_

Does your procedure generate a recursive or an iterative process? \_\_\_\_\_

**Problem 7 continued**

b) Write a procedure called `max-elts` which takes two lists and returns a list of the maximum elements at each corresponding location. For example,

```
(max-elts '(1 20 33) '(10 2 3))
```

should return the list

```
(10 20 33)
```

Your procedure should use the `combine-two-lists` procedure that you wrote above in part a. You may also find it helpful to know that scheme has a built-in procedure called `max`.

**Problem 8 (15 points)**

The food service department has asked us to help them write code that will help them plan their menus. First, we need to create a means for storing information about a recipe: each recipe record will contain the name of the item, the number of ingredients, and the number of servings that can be obtained from the recipe.

We define the constructor `make-recipe` as follows:

```
(define (make-recipe food-name num-ingredients num-servings)
  (list food-name (cons num-ingredients num-servings)))
```

Write the selectors `recipe-name`, `num-recipe-ingredients`, and `num-recipe-servings`.

To store all of the recipes we will use the following constructor:

```
(define make-recipe-list list)
```

Write the selectors `first-recipe` and `rest-recipes`.

**Problem 8 continued**

Write a procedure called `count-servings` which takes a `recipes-list` and returns a total of the number of servings that can be obtained by making all of the items in the recipe list. Be sure to use the constructors and selectors that you helped define on the previous page.

**Problem 9 (5 points)**

Given the following definition of `cons`, define `car` and `cdr`.

```
(define (cons x y)
  (lambda (z) (if (= z 1) x y)))
```