csc324 Quiz #2 Tuesday 01 February 2000

Duration: 15 minutes

Aids allowed: None

Family Name: ____________________ **Lecturer:** Diane Horton

First Name: ____________________

Student #: ____________________

Tutorial (circle one): A-F G-Lee Lef-Se Sf-Z

Grade: __________

csc324 Quiz #2 Tuesday 01 February 2000

1. What is the value of each of these expressions?

`( (lambda (a b) (cond ((> a b) (- a b)) (t (- b a)))) 3 7 )` Answer: `4`

`( (lambda (car cdr) (cons car cdr)) '(1 2 3) '(p q r) )` Answer: `((1 2 3) P Q R)`

2. Consider the following functions:

```
(defun sum (list)
    (cond ( (null list) 0 )
          ( (>= (sum (cdr list)) 0)
                 (sum (cdr list))
          )
          ( t -1 )
    )
)
```

What is the value of `(sum '(1 2 -5 -1 3))`? Answer: `0`

What is the big-oh time complexity of a call to `sum` with a list of length n, in the worst case?

Answer: $O(2^n)$
(See the next page for how you might derive this answer.)

3. Consider the following pair of functions:

```
(defun doit (list1 list2)
    (cond ( (null list1) nil )
          ( (help (car list1) list2)
              (doit (cdr list1) list2)
          )
          ( t  (cons (car list1) (doit (cdr list1) list2))
          )
    )
)

(defun help (x y)
    (cond ( (null y) nil )
          ( (equal x (car y))  t )
          ( t  (help x (cdr y))  )
    )
)
```

What is the value of `(doit '(a b c d e) '(b a e))`? Answer: `(C D)`

What is the value of `(doit '(1 2) '(8 7 6 5 4 3 2 1))`? Answer: `NIL`

What is the value of `(doit '(p q r) '(s t u))`? Answer: `(P Q R)`

Let $T(n)$ be the time to execute a call to `sum list` on a list of length n, in the worst case.

On a list of length 0, `null` is called and 0 is returned, which in total is $O(1)$. It's a little easier to deal with the recurrence relation if we take the big-oh out. We can do this easily: $T(0) = c_1$ for some constant $c_1 \geq 1$.

On a list of any length greater than 0, in the worst case the second `cond` clause succeeds. We end up calling `null` once, take the `cdr` of the list twice, and do a `>=` comparison (which in total is $O(n)$); and we also call `sum` twice on that `cdr` (which in total takes time $2 \times T(n-1)$). Taking the big-oh out, we can say that $T(n) = c_2 + 2 \times T(n-1)$.

To recap, we have defined the following recurrence relation: For some c_1 and $c_2 \geq 1$,

$$\begin{aligned} &T(0) = c_1 \\ &T(n) = c_2 + 2 \times T(n-1), \text{ for all } n > 0. \end{aligned}$$

To solve the recurrence relation, we can make a table of values and look for a pattern.

n	$T(n)$	
0	c_1	
1	$c_2 + 2 \times (c_1)$	
2	$c_2 + 2 \times (c_2 + 2c_1)$	$= c_2 + 2c_2 + 4c_1$
3	$c_2 + 2 \times (c_2 + 2c_2 + 4c_1)$	$= c_2 + 2c_2 + 4c_2 + 8c_1$
4	$c_2 + 2 \times (c_2 + 2c_2 + 4c_2 + 8c_1)$	$= c_2 + 2c_2 + 4c_2 + 8c_2 + 16c_1$

The pattern is

$$\begin{aligned} T(n) &= c_2 + 2c_2 + 4c_2 + 8c_2 + \cdots + 2^{n-1}c_2 + 2^n c_1 \\ &= c_2 \times \sum_{i=0}^{n-1} 2^i + 2^n c_1 \end{aligned}$$

and this is $O(2^n)$. As an exercise, you might try proving that this solution to the recurrence relation is correct.