University of Toronto
csc324— Principles of Programming Languages, Fall 1999

Midterm Test

Duration: 50 minutes

Aids allowed: None

Make sure that your examination booklet has 4 pages (not including this one). Write your answers in the spaces provided. Write legibly.

Surname: ______________________ **First name:** ______________________

Student #: ______________________

Tutor (circle one): Diana Ziau Michal Karczmarek Peeter Piegaze Anastasia Bezerianos

1.	________	/ 6
2.	________	/ 18
3.	________	/ 7
Total	________	/ 31

Question 1 [6 marks in total]

(a) [2 marks]
If we were to load the following Prolog database:

```
mother(petra, katie).
mother(katie, diane).
mother(diane, william).
grandmother(katie, william) :- mother(katie, X), mother(X, william).
```

and then ask `grandmother(petra, diane)`, Prolog would say `no`. Why? Circle the best answer.

1. No fact declares `grandmother(petra, diane)` to be true.
2. The facts and rules don't allow `grandmother(petra, diane)` to be inferred.
3. The grandmother rule cannot be properly expressed in first-order logic.
4. The grandmother rule cannot be properly expressed in Prolog.
5. `petra` should begin with an upper-case letter.

(b) [4 marks]
Suppose we use `prereq(x,y)` to mean that course `x` is a prerequisite for course `y`, and `has(s,c)` to mean that student `s` has taken course `c`. According to the following Prolog database:

```
prereq(csc108, csc148).
prereq(csc148, csc270).
canTake(Student, Course) :- prereq(Pre, Course), has(Student, Pre).
```

csc108 has no prerequisite. What happens if I ask prolog `canTake(fred, csc108)`?

`Prolog replies 'no'.`

Use logic to express a rule about courses without prerequisites that is missing from the database.

`Answer:`

$$\forall c, s, (\neg\exists p, \text{prereq(p, c)} \Rightarrow \text{canTake(s, c)}).$$

`or in other words,`
`For all c and s: If there is no p that is a prerequisite of c,`
`then s can take c.`

`Another solution:`

$$\forall c, (\neg\exists p, \text{prereq(p, c)} \Rightarrow \forall s, \text{canTake(s, c)}).$$

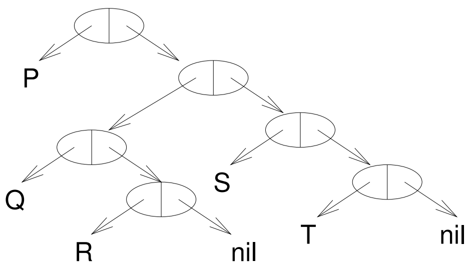
`In other words,`
`For all c: If there is no p that is a prerequisite of c,`
`then for all s, s can take c.`

Question 2 [18 marks in total]

(a) [3 marks]
Suppose we are running Lisp, and variables `list1` and `list2` have not been mentioned yet. Draw a tree showing Lisp's internal representation for `list1` after the expression

`(setq list1 '(P (Q R) S T))`

is evaluated.



(b) [3 marks]
Suppose that the expression `(setq list2 (cons 'A list1))` is then evaluated. How many pieces of new memory are allocated?

```
Two.
```

What is each piece for?

```
One is for the cons cell that connects A to list1
and the other is for the new global variable list2.
```

(c) [3 marks]
Consider the following Lisp function:

```
(defun foo (a b)
    (setq temp (+ a 3))
    (cons temp b)
)
```

For each variable, circle one phrase to indicate whether it is static, stack-dynamic, or heap-dynamic.

`a`	static	$\boxed{\text{stack-dynamic}}$	heap-dynamic
`b`	static	$\boxed{\text{stack-dynamic}}$	heap-dynamic
`temp`	static	stack-dynamic	$\boxed{\text{heap-dynamic}}$

(d) [9 marks]
Consider the following program, written in pseudo-code:

```
function c (int n) {
    print "In c"
    print n, y
}

function b () {
    print "In b"
    int y = -3
    print n, y
    c(41)
}

function a (int p) {
    print "In a"
    int n = 1000
    print n, y
    b()
}

main () {
    int x = 12
    int y = 99
    a(x)
}
```

If the language used static scope, would this program compile and run? Circle one answer.
Yes $\boxed{\text{No}}$
If yes, what output would the program produce? If no, why not?

```
Variables are used out of their scope
(y inside function c, y and n inside function b, and y inside function a).
```

If the language used dynamic scope, would this program compile and run? Circle one answer.
$\boxed{\text{Yes}}$ No
If yes, what output would the program produce? If no, why not?

```
In a
1000  99
In b
1000  -3
In c
41 -3
```

Question 3 [7 marks in total]

(a) [3 marks]
Complete the Lisp function below without using recursion.

```
; Returns the average of the integers in ‘list’.
; Precondition: ‘list’ is a list of integers, with length >= 1.
(defun avg (list)

     (/ (apply ’+ list) (length list))
)
```

(b) [2 marks]
The `samelist` function below does not work. Write a call to this function that meets the preconditions yet causes a crash. Hint: This is not about `eq` vs `equal`.

```
(samelist ‘(A (B C)) ‘(A B C))
```

```
Explanation (not required for full marks):
      A crash is caused by any pair of lists such that an element of the first
      list is itself a list, but the corresponding element of the second list
      is an atom.  In our example, the second element the first list is the list
      (B C), whereas the second element of the second list is the atom B.
      This leads to a recursive call (samelist ‘(B C) ‘B) which violates
      a precondition of the function.  That call then crashes when it
      evaluates (car y), because y is not a list.
```

(c) [2 marks]
Circle the the very last function call that is executed — the one that directly causes the crash.

```
; Returns true (non-nil) iff x and y are the same list.
; Examples: (samelist ‘(a (b c)) ‘(a (b c))) returns t
;           (samelist ‘(a b c) ‘(a (b c))) returns nil.
; Precondition: ‘x’ is a list and ‘y’ is a list.

(defun samelist (x y)
     (cond ( (null x) (null y) )
                                              -----------
            ( (atom (car x)) (and (atom | (car y) | )
                                              -----------
                                  (eq (car x) (car y))
                                  (samelist (cdr x) (cdr y))
                             )
            )
            ; (car x) must be a list.
            ( t (and (samelist (car x) (car y)) (samelist (cdr x) (cdr y))) )
     )
)
```

END OF TEST