CS150: Exam 2
Due: Mon Apr 20th by 3:30pm (in class) or by 6:30pm (OLS 219)

UVA ID: ____

Directions

Please be honorable. As long as everyone follows the honor code, take home exams benefit everyone. You are on your honor to follow the letter and spirit of these directions. You do not need to scribble a pledge on your exam to convince us you are honorable, but if you cheat please let us know so it does not unfairly impact other students. Please don't cheat.

Work alone. You may not discuss these problems or anything related to the material covered by this exam with anyone except for the course staff between receiving this exam and class Monday.

Open resources. You may use any books you want, lecture notes, slides, your notes, and problem sets. You may also use DrScheme or Python, but it is not necessary to do this. You may also use external non-human sources including books and web sites. If you use anything other than the course books, slides, and notes, cite what you used. You may not obtain any help from other humans other than the course staff.

Answer well. Answer all questions 1-9 (question 0 is writing your UVA ID on this page and the next, as well as *stapling the exam*, which hopefully everyone will receive full credit for), and optionally answer subsequent questions.

You may either: (1) print out this exam and write your answers on it or (2) write your answers directly into the provided Word template and print the result out. Whichever one you choose, you must turn in your answers printed on paper and they must be clear enough for us to read and understand. You should not need more space than is provided to write good answers, but if you want more space you may attach extra sheets. If you do, make sure they are clearly marked. Staple your completed exam before turning it in.

The questions are not necessarily in order of increasing difficulty, so if you get stuck on one question you should continue on to the next question. There is no time limit on this exam, but it should not take a well-prepared student more than a few hours to complete. It may take you longer, though, so please do not delay starting the exam. There is no valid excuse (other than a medical or personal emergency) for running out of time on this exam.

Full credit depends on the clarity and elegance of your answer, not just correctness. Your answers should be as short and simple as possible, but not simpler. Your programs will be judged for correctness, clarity and elegance, but you will not lose points for trivial errors (such as missing a closing parenthesis).

NOTE: You will receive 30% of the credit for a (sub)question if you leave it totally blank (and thus do not waste our time). A wrong but present answer might receive as little as 0% of the credit. There is no need to guess.

UVA ID:

____

Scores

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**EC**	**Tot**
10	10	10	10	10	10	10	10	10	10	0	100

Networks

1a. Define a `latency` procedure that takes one input, a list of lists of bit arrival times, and produces as output a number representing the latency of the communication. For example, imagine that we send three bits across four semaphore stations. The first bit reaches the first station at 1 second, the second station at 3 seconds, the third station at 5 seconds, and the last station at 7 seconds, as shown (bits two and three are not described in text, but are shown below):

```
> (define bit-one   (list 1 3 5 7))
> (define bit-two   (list 4 6 8 10))
> (define bit-three (list 5 6 10 12))
> (define bit-list  (list bit-one bit-two bit-three))
> (latency bit-list)
7
```

The input list will never be empty. Each individual bit arrival time list will never be empty and will consist of strictly-increasing numbers. All bit arrival time lists will have equal lengths. You may always use any procedure defined in class or the problem sets.

```
(define (latency input)
```

1b. Define a `bandwidth` procedure that takes one input, a list of lists of bit arrival times (exactly as above), and produces as output a number representing the total overall bandwidth of the entire connection. Continuing the example above:

```
> (bandwidth bit-list)
1/4
```

```
(define (bandwidth input)
```

Mutation and Databases

2a. Consider the following definition for a while loop, as per Chapter 10.4:

```
> (define (while test body)
       (if (test)
            (begin (body) (while test body))
            (void)))    ;;; no return value
```

Define a procedure `list-map!` that takes two parameters: a worker function and a list. Your `list-map!` procedure should modify the list in place, replacing each element with the result of applying the worker function to it. Your `list-map!` procedure must use `while`, and may not be directly recursive (i.e., all recursion must happen through a single call to `while`).

```
> (define x 2)
> (while    (lambda () (> x 0))
            (lambda () (display x) (set! x (- x 1))))
2 1
> x
0
> (define mylist (list 1 2 3 4))
> (list-map! (lambda (x) (* x x)) mylist)
> mylist
(1 4 9 16)
```

```
(define (list-map! workfun lst)
```

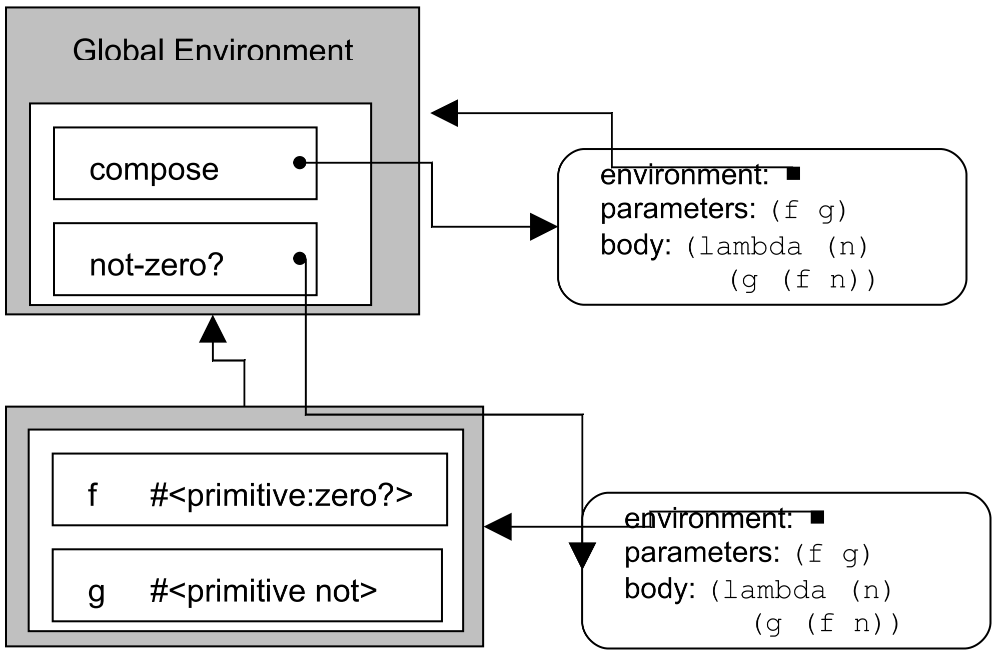
2b. Consider the following database table named `CSBooks`. Give a SQL `select` query that returns all book titles written in the 1990s that cost less than 14.00, sorted by author name. Be as generic as possible: support tables with different entries than the example one.

BookID	Title	Author	Price	Publisher	Year
1	The Mind's I	Hofstadter	18.95	Bantam	1985
2	GEB	Hofstadter	19.95	Basic	1979
3	Cryptonomicon	Stephenson	14.25	Perennial	1999
4	The Code Book	Singh	14.00	Anchor	2000
5	Snow Crash	Stephenson	13.75	Bantam	1992

```
SELECT
```

Global Environments

3. Consider the environment shown below (as in question 1, assume all the usual primitives are defined in the global environment, but not shown; the notation #<primitive:zero?> denotes the primitive procedure zero?):



Provide a sequence of Scheme expressions such that evaluating the sequence of expressions produces the environment shown above. Hint: give two definitions.

Object-Oriented Programming

4. Consider the following PS6-style definitions:

```
(define (make-object name1)
  (lambda (message1)
    (case message1
      ((class) 'object)
      ((name) name1)
      (else no-method))))

(define (make-person name2)
  (let ((super (make-object name2))
        (possessions '())
        (restlessness 0.0))
    (lambda (message2)
      (case message2
        ((class) 'person)
        (else (super message2))))))
```

Draw the global environment while executing:

```
      (define turing (make-person 'turing))
      (turing 'name)
```

Hint: be sure to draw all frames that might be created as a result of function evaluations.

Computability

5. Is the *Contains-Cross-Site-Scripting-Vulnerability Problem* described below computable or uncomputable? Your answer should include a convincing argument why it is correct.

Input: *P*, a specification (all the code and html files) for a dynamic web application.

Output: If *P* contains a cross-site-scripting vulnerability, output **True**. Otherwise, output **False**.

As demonstrated in class, a cross-site-scripting vulnerability is an opportunity an attacker can exploit to get their own script running on a web page generated by the web application.

6. Is the *Remove-Cross-Site-Scripting-Vulnerabilities Problem* described below computable or uncomputable? Your answer should include a convincing argument why it is correct.

Input:	*P*, a specification (all the code and html files) for a dynamic web application.
Output:	*P′*, a specification for a dynamic web application. On inputs that are not cross-site-scripting attacks, *P′* behaves identically to *P*. On inputs that are cross-site-scripting attacks, *P′* ignores the attack input and displays a warning page.

Interpreters and Asymptotic Running Time

7a. For the next two questions, you are given a procedure definition. Your answer should describe its asymptotic running time when evaluated using (a) Charme, and (b) MemoCharme (the language defined at the end of PS7), and (c) LazyCharme. You may assume the Python dictionary type provides lookups with running time in O(1). Your answers should include a clear supporting argument, and define all variables you use in your answer.

```
(define duplicitous
    (lambda (a)
        (cond
            ((> a 0) (+ (duplicitous (- a 1))
                        (duplicitous (- a 1))))
            ((zero? a) 1))))
```

(a) Running time in Charme:

(b) Running time in MemoCharme:

(c) Running time in LazyCharme:

7b.

```
(define temeritous
    (lambda (a b)
        (cond
            ((> a 0) (temeritous (- a 1)
                                 (temeritous (+ a 1) b)))
            ((zero? a) 2))))
```

(a) Running time in Charme:

(b) Running time in MemoCharme:

(c) Running time in LazyCharme:

Static Type Checking

8. (as promised, Exercise 14.2) Define the `typeConditional(expr, env)` procedure that checks the type of a conditional expression. It should check that all of the predicate expressions evaluate to a Boolean value. In order for a conditional expression to be type correct, the consequent expressions of each clause produce values of the same type. The type of a conditional expression is the type of all of the consequent expressions. (You may assume the StaticCharme interpreter described in Chapter 14.)

9. (based on Exercise 14.3) A stronger type checker would require that at least one of the conditional predicates must evaluate to a true value. Otherwise, the conditional expression does not have the required type (instead, it produces a run-time error). Either define a `typeConditional` procedure that implements this stronger typing rule, or explain convincingly why it is impossible to do so.

10a. Do you feel your performance on this exam will fairly reflect your understanding of the course material so far? If not, explain why.

10b. (2 points extra credit.) Indicate that you went to Scott Aaronson's lecture on Quantum Computing and give one thing you learned there.

10c. (2 points extra credit.) Indicate that you completed Kinga Doboyli's web fault severity survey.

10d. (2 points extra credit.) Indicate that you went to the Bill Wulf and Anita Jones fireside chat and give one thing that you learned there.

End of Exam