cs150: Final Exam

Name: _____

Due: Monday, 7 May at 4:55pm

Turn in your completed exam to David Evans or to Brenda Perkins in the CS front office.

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 1-10. (Please also answer your name correctly, but there are no points for this on the final.)

You may either: (1) write your answers on this exam or (2) type and write your answers into the Word document template (*http://www.cs.virginia.edu/cs150/exams/final/final.doc*). Whichever 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 clearly-marked extra sheets.

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 complete.

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).

Score: _____

Procedures

Questions 1 and 2 ask you to define procedures that solve specified problems. For each question, you may use any programming language that has been mentioned in CS150 (Scheme, Python, Charme, LazyCharme, StaticCharme, Fortran, FP, FL, Java, C, C++, PHP, LISP, or JavaScript). If you use a language other than Scheme or Python, you should clearly state in your answer which language you are using.

1. Define a procedure, **`xorlist`**, that takes as input a list, and outputs the result of XOR-ing all elements in the list. The result should be true if the list contains an odd number of true values, and false if the list contains an even number of true values. You may assume a procedure, **`xor`**, is defined that takes two inputs and outputs the exclusive or of those inputs.

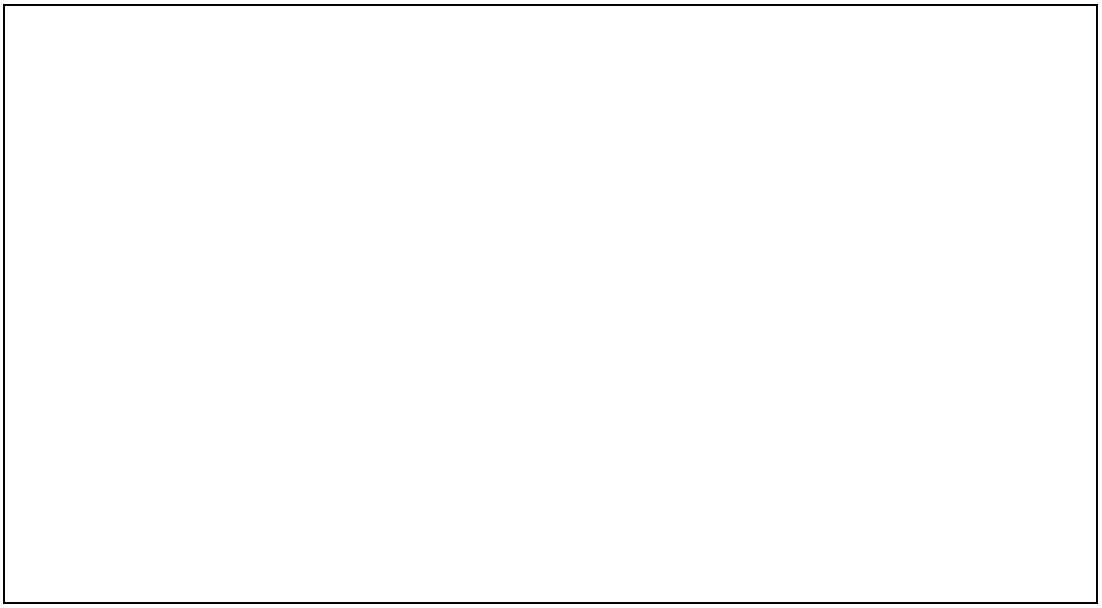
For example,

Python:

```
>>> xorlist([True])
True
>>> xorlist([])
False
>>> xorlist([True, False, True, False, True])
True
```

Scheme:

```
> (xorlist (list #t))
#t
> (xorlist null)
#f
> (xorlist (list #t #f #t #f #t))
#t
```



2. Define a procedure, **`findlast`**, that takes two inputs, a list and a predicate procedure, and outputs the last element in the list for which the predicate procedure applied to that element evaluates to true. If no element in the list satisfies the predicate procedure, **`findlast`** should produce **`null`** (in Scheme) or **`None`** (in Python). For full credit, your procedure's running time should be in $O(n)$ where n is the number of elements in the input list.

For example,

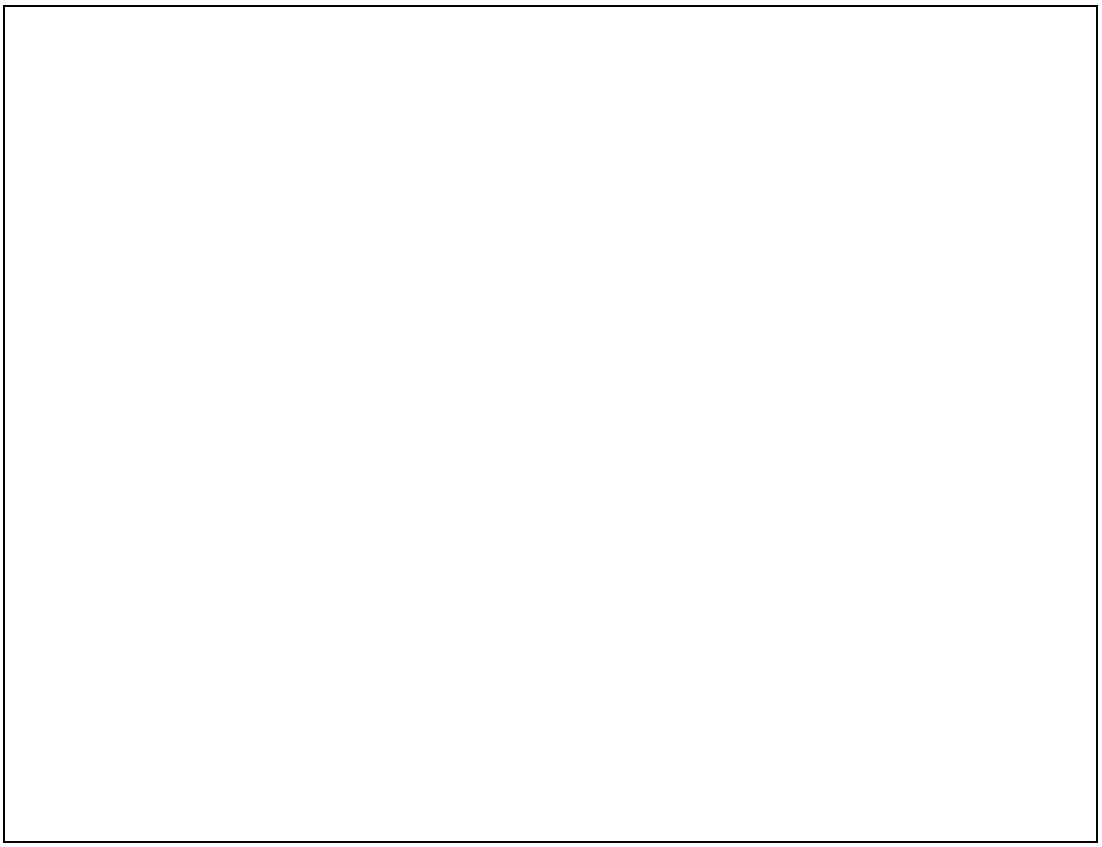
Python:

```
>>> def iseven(n): return n%2 == 0
>>> findlast([1,2,3], iseven)
2
>>> findlast([1,2,3,4], iseven)
4
>>> findlast([1,3], iseven)
```

evaluates to **`None`** (which doesn't print out in Python)

Scheme:

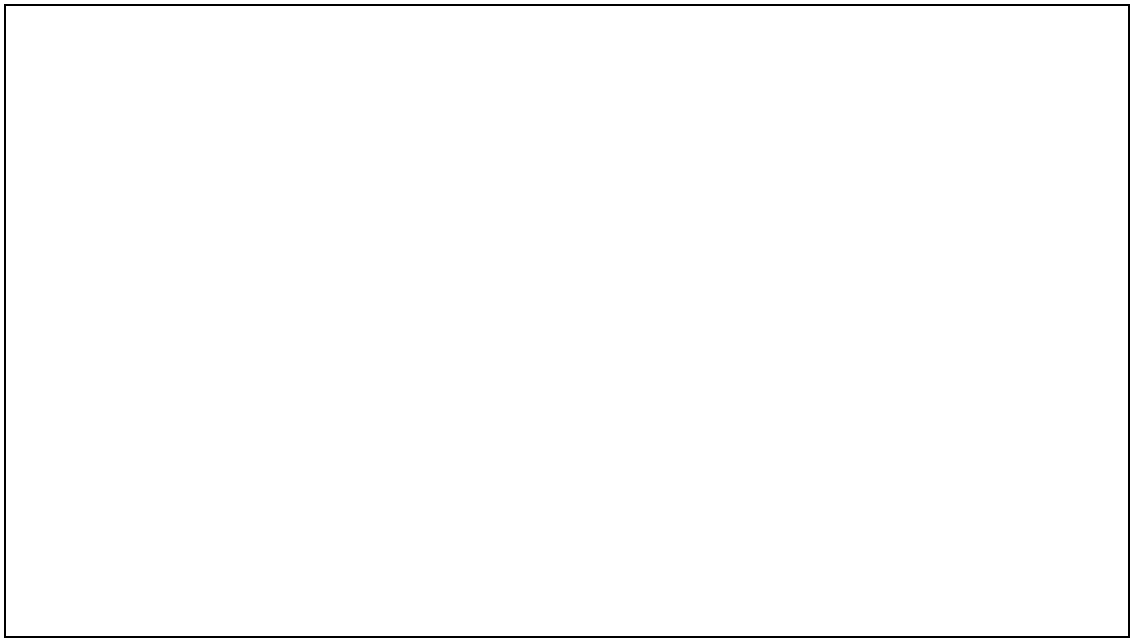
```
> (findlast (list 1 2 3) even?)
2
> (findlast (list 1 2 3 4) even?)
4
> (findlast (list 1 3) even?)
()
```



Running Time Analysis

3. What is the asymptotic running time of the **`findmiddle`** procedure defined below? Your answer should define all variables it contains and clearly state all assumptions you use.

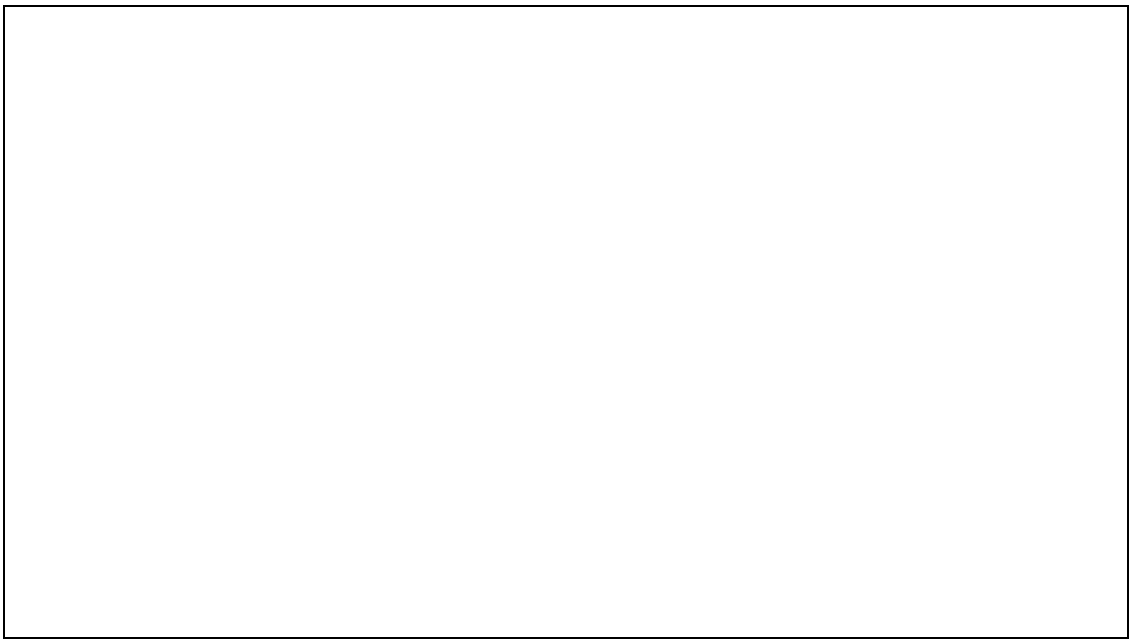
```
(define (find-middle lst)
  (define (find-middle-helper lst n)
    (if (= n 0)
        (car lst)
        (find-middle-helper (cdr lst) (- n 1))))
  (find-middle-helper lst (floor (/ (length lst) 2))))
```



4. What is the asymptotic running time of the **`intersect`** procedure defined below? Your answer should define all variables it contains and clearly state all assumptions you use.

```
(define (contains? lst val)
  (if (null? lst)
      #f
      (if (eq? (car lst) val)
          #t
          (contains? (cdr lst) val))))

(define (intersect lst1 lst2)
  (if (null? lst1) null
      (if (contains? lst2 (car lst1))
          (cons (car lst1) (intersect (cdr lst1) lst2))
          (intersect (cdr lst1) lst2))))
```



Object-Oriented Programming

5. (Exercise 10.3 from the Course Book) Define a new subclass of **`poscounter`** where the increment for each **`next!`** method application is a parameter to the constructor procedure. For example, **`(make-var-counter 0.1)`** would produce a counter object whose counter has value **`0.1`** after one invocation of the **`next!`** method. (You should assume all definitions from Chapter 10.)

Computability/Turing Machines

6. Is the *Writes-Hash Problem* described below computable or uncomputable? Your answer should include a convincing argument why it is correct.

Input: A specification of a Turing Machine, *T*, including its tape, using the Turing Machine description grammar from Lecture 37.

Output: Outputs true if running *T* would ever write a **#** symbol on the tape; otherwise, outputs false.

Lambda Calculus

7. Define a Lambda Calculus term, **sub**, that corresponds to subtraction. You may assume all the definitions from Lecture 40. Your **sub** definition should satisfy the following properties:

sub 1 zero $\Rightarrow$ **1**
sub 1 1 $\Rightarrow$ **zero**
sub 2 1 $\Rightarrow$ **1**
sub 2 zero $\Rightarrow$ **2**
sub 5 2 $\Rightarrow$ **3**

Interpreters

The next two questions as you to modify the StaticCharme interpreter to support the **`begin`** special form. You do not need to modify the interpreter, but if you want to try out your solution you can download the StaticCharme interpreter from *http://www.cs.virginia.edu/cs150/final/staticcharme.zip*. (This has been updated from the version provided in the Exam 2 comments to include the changes to **`meval`** and **`typecheck`** necessary to support begin expressions, with stub procedures for your answers to question 8 and 9.)

8. Define the **`evalBegin`** procedure to implement the evaluation rule for **`begin`**. The StaticCharme begin expression should have the same evaluation rule as the Scheme begin expression.

9. Define the **`typeBegin`** procedure to implement the type checking rule for **`begin`**. The type of a **`begin`** expression is an error type if the expression has no subexpressions, or if any of the subexpressions are mistyped. Otherwise, the type is the type of the last subexpression.

For example,

```
StaticCharme> (begin (+ 3 #t) 4)
Error: Parameter type mismatch: expected (Number Number), given
(Number Boolean)
StaticCharme> (begin )
Error: No expressions in begin
StaticCharme> (begin (+ 3 3) #t #f (* (+ 7 8) 10))
150
```

10. a) Explain why it does not make any sense to actually add **`begin`** to StaticCharme, unless additional special forms or primitives are also added.

b) Explain why LazyCharme does not need a **`begin`** special form (even if additional primitives and special forms were added to LazyCharme).

These questions are optional and worth no credit, but we appreciate your answers.

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

In determining your final grade for the course, is there anything you think I should take into account that would not be readily apparent from your recorded performance on the problem sets, quizzes, and exams?

Please remember to submit your **Official Course Evaluation** and return your completed **Course Improvement Survey** when you turn in this exam.

Enjoy your summer and thanks for your contributions to the course!

End of Exam