University of Virginia **Out: 9 October 2011**
cs1120: Introduction of Computing **Due: 11:01 am, Wednesday, 14 October**
Explorations in Language, Logic, and Machines

Exam 1

Name: ______________________________ **UVa Email ID:** __ __ __ __ __ __

Directions

Due: 11:01am, Wednesday, 14 October 2011. No late exams will be accepted without prior arrangement or an extraordinarily good story (that should involve either a hospital or a jail cell).

Work alone. Between receiving this exam and turning it in at class Wednesday, you may not discuss these problems or anything related to this exam with anyone other than the course staff.

Open resources. You may use any books you want, lecture notes, slides, your notes, and problem sets, including any materials posted on or linked from the course website. **You may not use DrRacket, or any other Scheme interpreter**. You may also use external non-human sources including books and web sites. If you use anything other than the course book, slides, and notes, you must cite what you used clearly in your answer. **You may not obtain any help from other humans** other than the course staff (who will only answer clarification questions).

Answer well. Answer all of the graded questions and the optional, ungraded questions on the last page. A "full credit" answer for each question is worth 10 points (but it is possible to get more than 10 points for especially elegant and insightful answers). The exam has more than 100 possible points, but a score of 100 is considered a "maximum" score on this exam.

Your answers must be clear enough for us to read and understand. If you do not use our provided print-out, your print-out should use only the front sides of pages and should be printed well enough to be easily read.

You should not need more space than is provided to write good answers, but if you need more you may use the backs or attach extra sheets. If you do, make sure the answers are clearly marked.

The questions are not necessarily in order of increasing difficulty. There is no time limit on this exam, but it should not take a well-prepared student more than two hours to complete. It may take you longer, though, so please do not delay starting the 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 answers will be judged for correctness, clarity and elegance, but you will not lose points for trivial errors (such as missing a closing parenthesis).

Pledge: ____
Sign here to indicate that you read, agreed to, and followed all of the directions here in addition to the Course Pledge.

Language

1. (a) Give a BNF grammar that produces the language of DNA sequences. Your grammar should produce all strings of **zero or more nucleotides**, where the nucleotides are represented by elements from the set { **A**, **C**, **G**, **T** }, and no other strings.

   (b) Give a BNF grammar that produces the language of DNA sequences that contain **one or more full codons**. Each codon is a sequence of three nucleotides. For example, your language should contain "**ACCGACTAA**" but should not contain "**ACGA**" or "" (the empty string).

2. For each item below, check the one answer that best characterizes the entire Scheme fragment shown. (Note: this is not asking you about what it evaluates to!)

a. **(+ 0)**

___ Application Expression
___ Primitive Expression
___ Procedure Expression
___ Special Form

b. **((lambda (v) v) 3)**

___ Application Expression
___ Primitive Expression
___ Procedure Expression
___ Special Form

c. **(lambda (v) +)**

___ Application Expression
___ Primitive Expression
___ Procedure Expression
___ Special Form

d. **(if true false true)**

___ Application Expression
___ Primitive Expression
___ Procedure Expression
___ Special Form

e. **((if true (lambda (a) false) (lambda (b) true)) 3)**

___ Application Expression
___ Primitive Expression
___ Procedure Expression
___ Special Form

Procedures

3. Define a procedure, **any-matches**, that takes as input three numbers and outputs **true** if any two of the numbers are equal, and outputs **false** otherwise. For example,
   (any-matches 3 7 3) should evaluate to **true**
   (any-matches 3 4 5) should evaluate to **false**
   (any-matches 7 3 7) should evaluate to **true**

4. Define a procedure, **is-composite?**, that takes as input a natural number, and outputs **true** if that number is composite, and outputs **false** otherwise. A natural number, *n*, is composite if it is divisible by some number that is greater than 1 and less than *n*. You do not need to worry about efficiency in your solution – a simple, slow, and correct procedure is worth full credit.

   You may use the **is-divisible?** procedure defined below in your **is-composite?** definition. The **is-divisible?** procedure takes two inputs, and evaluates to **true** if the first input is divisible by the second input, and **false** otherwise.

   ```
   (define (is-divisible? v d) (= (modulo v d) 0))
   ```

5. Define a procedure, **all-positive?**, that takes as input a list. The output should be true if the all the elements in the input list are positive; otherwise, the output should be false. For example,

    (all-positive? (list 2 4 6 8)) should evaluate to **true**
    (all-positive? (list 2 0)) should evaluate to **false**
    (all-positive? null)) should evaluate to **true**

6. Define a procedure, **is-pure?**, that takes as input a list and a test procedure. The output should be true if the result of applying the test procedure to each element in the list is true; otherwise the output should be false. For example,

    (is-pure? even? (list 2 4 6 8)) should evaluate to **true**
    (is-pure? even? (list 2 4 6 8 9)) should evaluate to **false**
    (is-pure? (lambda (v) false) null)) should evaluate to **true**

7. Define a procedure, **factors**, that takes as input a number and outputs a list containing all the non-trivial factors of that number. A natural number *b* is a factor of *a* if *a* is divisible by *b*. The trivial factors (1 and the input number) should not be included. You may use the **is-divisible?** procedure from question 4 in your definition. For example,

    (factors 6) should evaluate to the list **(2 3)**
    (factors 7) should evaluate to the empty list
    (factors 1120) should evaluate to the list
    (2 4 5 7 8 10 14 16 20 28 32 35 40 56 70 80 112 140 160 224 280 560)

    Hint: define a **factors-helper** procedure and use it do define **factors** as:

    ```
    (define (factors n)
      (list-reverse (factors-helper (- n 1) n)))
    ```

    (If you use this, it is only necessary to show the definition of **factors-helper**.)

8. Lefty O'Doul proposes replacing the standard *List* structure, with a new structure he calls a *Tsil* (the t is silent, of course) defined as:

   A *Tsil* is either (1) **llun** or (2) a Pair whose first cell is a *Tsil*.

   where **llun** is a new special built-in value (analogous to **null**), and the built-in procedure **llun?** (analogous to **null?**) that takes one value as input and outputs **true** if that value is **llun**, and **false** otherwise.

   Define a procedure, **tsil-map**, analogous to **list-map**, that takes as input a procedure and a *Tsil*, and produces as output a *Tsil* that contains as elements the result of applying the input procedure to each element of the input *Tsil* in the same order.

Machines

9. Design a Turing Machine that takes as input a tape in the form *a*=*b*# where *a* and *b* are sequences of zero or more symbols where each symbol is either **0** or **1**. The output should be **#1** (this can be anywhere on the tape, but there should be exactly one **#** on the output tape) if *a* and *b* are exact matches, and **#0** otherwise. Your answer should first describe in English (or Schemish) a high-level description of how your machine works. Then, you should provide a precise description of your machine, either as a complete list of transition rules or as a diagram.

Tandem Repeats

A tandem repeat in a genome is a sequence of nucleotides that repeats two or more times consecutively. Tandem repeats are common throughout the genome, and have many important uses including genetic fingerprinting (this is what is used in DNA tests to identify criminals, since the number of repetitions for certain sequences is highly variable across humans and easy to detect) and diagnosing diseases (for example, in a normal FMR-1 gene the sequence CGG repeats 6-54 times consecutively, but repeats over 200 times in patients with a form of autism). In questions 10 and 11, you will define procedures for counting tandem repeats.

10. Define a procedure, **contains-matching-prefix**, that takes as input two lists, *p* and *s*. The output should be **true** if the sequence of elements at the beginning of *p* exactly matches the complete sequence of elements in *s*. Otherwise, the output should be **false**.

    For example,
    (contains-matching-prefix (list 1 2 3 4) (list 1)) should evaluate to **true**
    (contains-matching-prefix (list 1 2 3 4) (list 1 2 4)) should evaluate to **false**
    (contains-matching-prefix (list 1 2) (list 1 2)) should evaluate to **true**
    (contains-matching-prefix (list 1 2) (list 1 2 3)) should evaluate to **false**
    (contains-matching-prefix (list 1 2) null) should evaluate to **true**

11. Define a procedure, **count-tandem-repeats**, that takes as input a list, *p*, and a number, *n*. The output should be the total number of times the first *n* elements of *p* are consecutively repeated at the beginning of *p*. Repetitions may not overlap.

    For example,
    (count-tandem-repeats (list 1 2 1 2 1 3) 2) should evaluate to **2**
    (count-tandem-repeats (list 2 2 2 2 2 2) 2) should evaluate to **3**
    (count-tandem-repeats (list 1 2 3 1 2 1 2 3) 3) should evaluate to **1** [corrected]

You should feel free to define any helper procedures you think are useful, as well as to use any procedures defined in the course book or problem sets.

The questions on this page are ungraded. I do most appreciate your honest and thoughtful answers to them, though. Feel free to use additional space on the back if you wish.

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

13. How long did it take you to complete this exam?

14. What topics to you hope to see in the remainder of the course?

15. Do you trust your classmates to follow the honor expectations in this class? (Feel free to write comments instead or in addition to checking one or more of the options.)

    ___ Yes, I trust them completely.

    ___ I worry that there may be a few transgressions, but I believe the vast majority of the class is honorable and it is fair and beneficial to rely on this.

    ___ I think this class places too high a burden on students' honor, and there are enough dishonorable students that it is unfair on the honorable students.

    ___ I have reason to suspect that other students violated the honor policy on problem sets.

    ___ I have direct knowledge of other students violating the honor policy on problem sets.

    ___ I have reason to suspect that other students violated the honor policy on this exam.

    ___ I have direct knowledge of other students violating the honor policy on this exam.

Problem	Score	Notes
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
Total		