cs1120: Final Exam

Out: 7 December
Due: 4:59pm, Friday 11 December
[Drop off on paper at my office, Olsson 236A]

Name:

UVa Email ID:

Directions

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 when this exam is posted and 5pm Friday (with the exception of using a human with no significant computing knowledge in Question 10).

Open resources. You may use any books you want, lecture notes, slides, your notes, and problem sets. You may use DrScheme and Python and any other language interpreter you want. You may also use external non-human sources including books and web sites. If you use anything other than the course book, slides, notes, and interpreters cite what you used. Except for question 10, **you may not obtain any help from other humans** except for the course staff (who will only answer clarification questions). Note that Question 10 requires help from another human (for about 10-15 minutes) who does not have significant computing knowledge. So, don't wait until the last minute to line up your volunteer for Question 10.

Answer well. Answer as many of questions 1-10 as you can and the optional, ungraded question 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). You may either: (1) write your answers on this exam or (2) type and write your answers into the provided Word template:
http://www.cs.virginia.edu/cs1120/exams/finalexam/finalexam.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 in the marked boxes to write good answers, but if you want more space 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 programs will be judged for correctness, clarity and elegance, but you will not lose points for trivial errors (such as missing a closing parenthesis).

Pledge:

Scores:

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

Defining Procedures

For questions 1-3, provide a procedure with the described procedure. For each question, you may use either Scheme or Python for your procedure. If you are not confident your code is correct, it is also a good idea to include an English prose description of your procedure.

1. Define a procedure *count-positive* that takes a input a list of numbers and produces as output a number representing the number of positive numbers in the input list.

For example,

Scheme:	(*count-positive* (*list* 3 -2 4 0 12))
Python:	*count_positive*([3, -2, 4, 0, 12])

should evaluate to 3.

2. **(Exercise 4.8)** Define a procedure *find-maximum-epsilon* that takes as input a function *f*, a low range value *low*, a high range value *high*, and an increment *epsilon*, and produces as output the maximum value of *f* in the range between *low* and *high* at interval *epsilon*.

For example,

Scheme:	(*find-maximum-epsilon* (**lambda** (*x*) (* *x* (- 5.5 *x*))) 1 10 1)
Python:	*find_maximum_epsilon*(**lambda** *x*: *x* * (5.5 - *x*), 1, 10, 1)

should evaluate to 7.5.

3. Define a procedure *item-count* that takes as input a list of items, and outputs a list of <item, count> pairs indicating for each item that appears in the input list the number of times that item occurs.

For example,

Scheme: (*item-count* (*list* 'everything 'plain 'plain 'salt 'sesame 'everything))
Python: *item_count*(['everything', 'plain', 'plain', 'salt', 'sesame', 'everything'])

should evaluate to:

Scheme: (('everything . 2) ('plain . 2) ('salt . 1) ('sesame . 1))
Python: {'everything': 2, 'plain' : 2, 'salt': 1, 'sesame': 1}

(note that the order in which elements appear in the output doesn't matter).

Analyzing Procedures

4. What is the asymptotic running time of the *list-cruncher* procedure defined below:

```
(define (list-cruncher p)
   (if (null? p)
       null
       (if (odd? (car p) )
           (list-cruncher (cdr p))
           (cons (car p) (list-cruncher (cdr p))))))
```

For full credit, your answer must carefully define the meaning of any variables you use.

5. What is the worst-case asymptotic running time of the *disjoint* procedure defined below?

```
def disjoint(p1, p2):
    """Returns true if p1 and p2 are disjoint (that is, there is no common element in
       both lists.)"""
    for e1 in p1:
        for e2 in p2:
            if e1 == e2:
                return False
    return True
```

For full credit, your answer must carefully define the meaning of any variables you use in terms of the size of the input, and explain what the worst-case inputs are.

6. Suppose you have two correct sorting procedures, *sortA* and *sortB*. The *sortA* procedure has asymptotic running time in $\Theta(N^2)$ where N is the number of elements in the input list. The *sortB* has asymptotic running time in $O(N \log N)$ where N is the number of elements in the input list. For a given application, you need to sort a list of 1120 numbers. Which procedure is best for this application? (Either provide a clear argument of which procedure is best, or a convincing explanation of why you do not have enough information to answer the question.)

Computability

7. **(similar to Exercise 12.2)** Is the *Same-Result* problem described below computable or noncomputable? Provide a convincing argument supporting your answer.

Input: Descriptions of two Turing Machines, *M1* and *M2*

Output: If the result of running *M1* starting with an empty input tape is the same as the result of running *M2* starting with an empty input tape, output a 1 at the left edge of the tape. Otherwise, output a 0 at the left edge of the tape. Two Turing Machines are considered to produce the same result if either they both do not halt, or they both halt and leave the final tape with the same contents.

8. **(Challenging)** Is the *Moves-Past-Square-1120* problem described below computable or noncomputable? Provide a convincing argument supporting your answer.

Input: Description of a Turing Machine, *M*

Output: If running *M* starting with an empty input tape would ever move past the 1120th square on the input tape, output true; otherwise, output false. The input tape is infinite in one direction (to the right), and starts on the leftmost square. The squares are numbered starting from 0, so output should be true if M would ever go beyond the square 1120 squares to the right of the left edge of the tape.

Interpreters

9. The Charme interpreter from PS7/Chapter 11 does not define a **begin** special form. Extend the Charme interpreter to support a **begin** special for with the evaluation rule:

Evaluation Rule: Begin. To evaluate a begin expression,

(**begin** *Expression*$_1$ *Expression*$_2$ … *Expression*$_k$)

evaluate each subexpression in order from left to right. The value of the begin expression is the value of the last subexpression, *Expression*$_k$.

Define an *evalBegin* procedure that evaluates a begin expression. You may assume the expression passed into your procedure is a valid begin expression (that is, you do not need to include defensive error checking code in your procedure).

```
def evalBegin(expr, env):
```

Computing Concepts

For the last question, you will need another person. This must be someone who has *no significant computing knowledge*. In particular, they should have never written a computer program or taken a computing course. Henceforth, we will call this person the *student*. Your student will need to help you for about 10-15 minutes. It is best if you can find someone who is 5-13 years, but if you can't find someone young it is fine to use a UVa student as your student as long as she/he does not have any significant computing knowledge.

Your goal for this question is to pick some concept from this course and convey something important and interesting about it to your student. The concept you choose to explain may be anything you what that was covered in the class and has some intellectual value. Examples of possible concepts include recursive definitions, defining languages, computability, the Church-Turing thesis, or sorting algorithms, but you do not need to limit yourself to these examples.

(There are no answer boxes for these questions, but your answers should not be longer than about one page for each question. You can answer by typing in this document, or by including a separate page.)

10. a. Identify one major concept from this course, and describe in your own words what it is and what is interesting and important about it. Write a short plan for how you will introduce and describe this concept to your student, and what you hope your student will understand after your lesson. Feel free to include pictures or drawings, and anything else that you think will be helpful.

b. Find your student and explain the concept to her/him, following your plan from part a. Write a short summary of how things went. Your summary should include interesting questions your student asked and how you answered them, and a description of what you think your student understood at the end of the lesson.

11. (optional, ungraded) If there is anything else you feel I should know to fairly assign your final grade, please explain it here.

End of Exam. *Enjoy your winter break!*