

CS 3723: MIDTERM EXAM I

Name: Solutions

Please read each question carefully and answer it in the space provided.
There are ten questions. Each question is worth 10 points.

1. Postscript is Turing-Complete. Use this knowledge to answer the following questions:

- (a) What can be said about the class of functions that can be computed by a Postscript document compared to those that can be computed by a Turing-Machine?

A postscript document can be written that would compute any function that can be computed by a Turing Machine.

- (b) If Church-Turing Thesis is correct, what can be said more generally about the class of functions that can be computed?

A postscript document can be written that would compute any function that can be computed.

- (c) Since the Halting Problem is undecidable, what can be said about how long it can take to render (i.e., execute) arbitrary Postscript documents?

It may take an unbounded, possibly infinite, amount of time and this cannot be predicted in advance.

2. (a) Besides Postscript, list four other computer languages that are Turing Complete.

Examples include: C, Java, Lisp, ML, C#, Perl

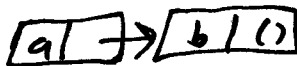
- (b) List three languages that are *not* Turing-Complete.

Examples include: HTML, SGML, PDF, SQL, CSS

3. What do each of the following Scheme expressions evaluate to? Write the result value. And if the result is a list, diagram the memory layout for the list.

- (a) `(cons 'a (cons 'b '()))`

(a b)



- (b) `(sqrt (+ (* 3 2) 10))`

4

- (c) `(+ (cdr (cons 1 2)) 7)`

9

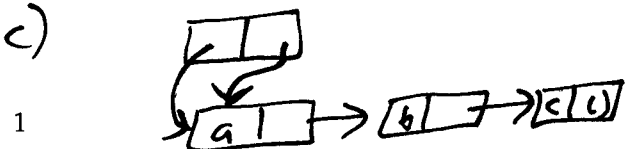
- (d) `(cons (car (cdr '(a 1 2))) '(+ 3 4))`

(1 + 3 4)



- (e) `((lambda (x) (cons x x)) '(a b c))`

((a b c) a b c)



4. Write a recursive Scheme function `count` that takes as parameters a symbol `s` and a list of symbols `l` and returns the number of occurrences of `s` in `l`. Assume `l` contains no nested sublists. Some example uses of `count` are:

```
(count 'a '(c a f e b a b e))  ⇒ 2
(count 'axis '(xylophone x axis)) ⇒ 1
(count 'x '(xylophone axel))   ⇒ 0
```

```
(define count
  (lambda (s l)
    (cond
      ((null? l) 0)
      ((eqv? (car l) s) (+ 1 (count s (cdr l))))
      (else (count s (cdr l)))))))
```

5. For each of the following λ -terms, indicate whether or not they are equivalent under α -renaming:

(a) xy $\not\equiv_{\alpha}$ yx

(b) $\lambda x.y$ $\not\equiv_{\alpha}$ $\lambda y.y$

(c) $\lambda x.\lambda y.xy$ \equiv_{α} $\lambda a.\lambda b.ab$

(d) $\lambda x.(\lambda y.y)y$ \equiv_{α} $\lambda x.(\lambda z.z)y$

(e) $(\lambda x.x)(\lambda y.y)$ \equiv_{α} $(\lambda x.x)(\lambda x.x)$

6. For each of the following λ -terms: If the λ -term is in its normal form, write "normal form." If the λ -term does not have a normal form, write "no normal form." Otherwise, show the step-by-step β -reduction of the λ -term to its normal form.

(a) x

normal form

(b) $\lambda x.x$

normal form

(c) $\lambda x.xy$

normal form

(d) $(\lambda x.x)a \rightarrow_{\beta} a$

(e) $((\lambda x.x)(abc))(def) \rightarrow_{\beta} (abc)(def)$

(f) $(\lambda x.x)(\lambda y.y) \rightarrow_{\beta} \lambda y.y$

(g) $(\lambda x.xx)(\lambda y.yy)$

no normal form

(h) $((\lambda x.\lambda y.yx)a)b \rightarrow_{\beta} (\lambda y.y a)b \rightarrow_{\beta} ba$

(i) $(\lambda x.\lambda y.xy)(\lambda z.a) \rightarrow_{\beta} \lambda y.(\lambda z.a)y \rightarrow_{\beta} \lambda y.a$

(j) $(\lambda x.x(xx))(\lambda z.a) \rightarrow_{\beta} (\lambda z.a)((\lambda z.a)(\lambda z.a)) \rightarrow_{\beta} a$

7. (a) What classes of formal languages make up the Chomsky hierarchy? Order them from *least to most general*.

regular, context-free, context-sensitive, recursively enumerable

(b) Which of these classes of languages is BNF used to describe?

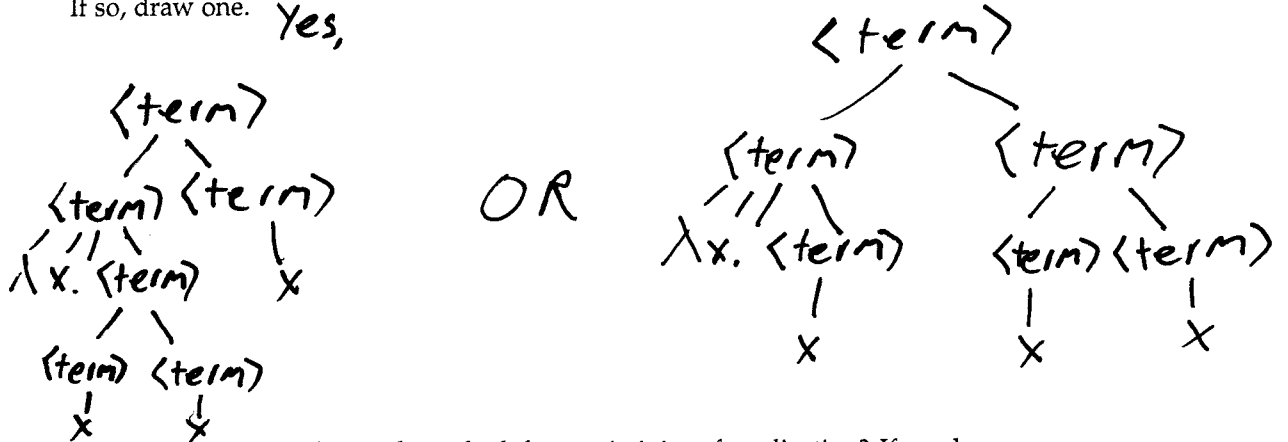
context-free

8. Consider the following simplified grammar for λ -terms:

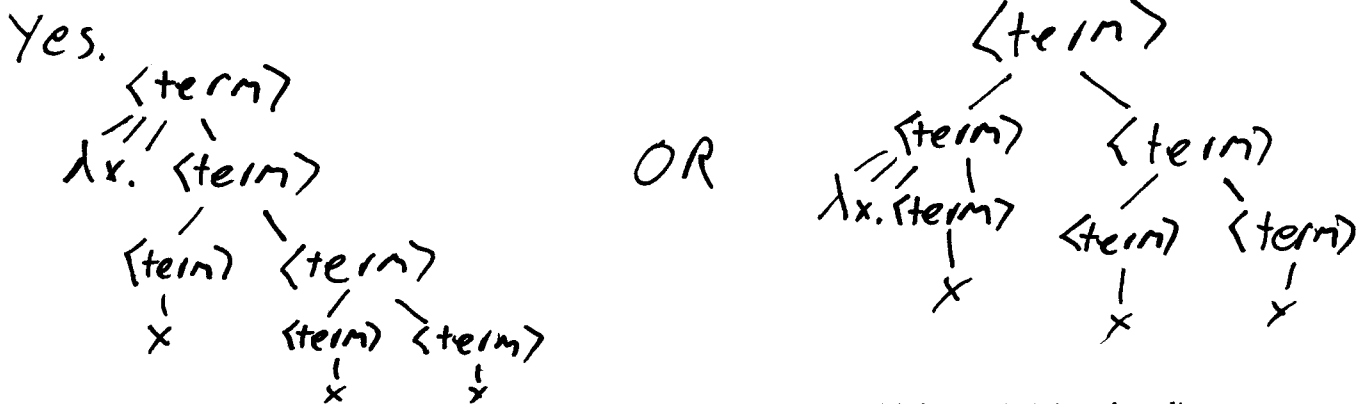
$\langle \text{term} \rangle ::= x \mid \langle \text{term} \rangle \langle \text{term} \rangle \mid \lambda x. \langle \text{term} \rangle$

And the wff $\lambda x. xxx$.

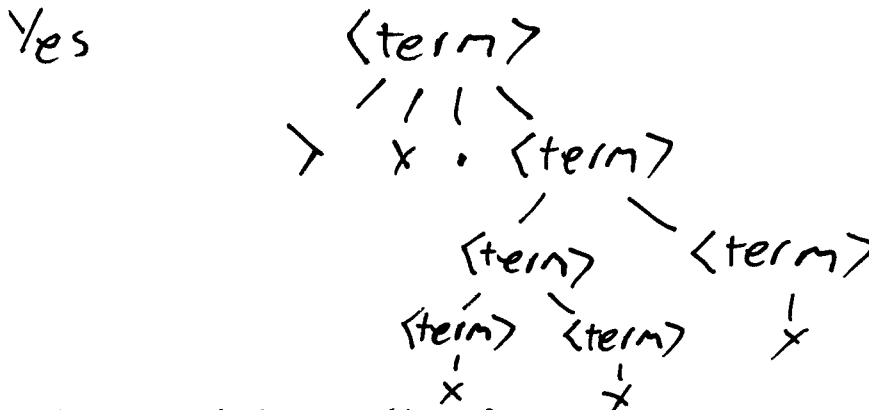
(a) Is there a parse tree that violates the higher precedence of application compared to abstraction? If so, draw one.



(b) Is there a parse tree that violates the left-associativity of application? If so, draw one.



(c) Is there a parse tree that is consistent with the higher precedence and left-associativity of application? If so, draw one.



(d) Is this grammar for λ -terms ambiguous?

Yes.

9. (a) List the six phases of a compiler that are described in Mitchell.

1. Lexical Analyzer
2. syntax Analyzer
3. Semantic Analyzer
4. Intermediate Code Generator
5. Code Optimizer
6. Code Generator

(b) Briefly explain what the Code Optimizer does.

It processes the intermediate representation, transforming it in various ways (e.g., removing unnecessary code) that make the resulting program more efficient without changing the program's semantics.

10. Answer the following questions about compilation and interpretation:

(a) What is a compiler?

A program that translates some other program written in a source programming language into instructions for some hardware machine.

(b) What is an interpreter?

A program that executes another program by simulating the language that program is written in by directly performing the tasks required by the statements in that program.

(c) Will a program usually run fastest as compiled machine-code or by being interpreted?

Compiled machine code

(d) How does a JIT compiler differ from a traditional compiler such as gcc?

It compiles parts of a program (e.g., methods) as needed while the program is running under the management of some virtual machine.