Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CSI 3120, Fall 2016, midterm exam A
October 20, 2016, 08:30 – 09:50
Lecturer: Dr. Rafael Falcon

Last name(s) First name(s) Student number

....

Notes

1. This is a closed-book exam. **No electronic devices are allowed**. You can have one double-sided *handwritten* page of notes, US Letter size (8.5 by 11 inches); no magnifying glass, please. No books and no other papers are permitted; use this questionnaire for rough work, including the spare page. If you have a bilingual English-to-X dictionary, you can use it, but only under supervision.
2. Write comments and assumptions to get partial marks.
3. Please attach the cheat sheet to the exam when you hand it in.
4. Beware that poor handwriting could affect grades.
5. Do not remove the staple holding the examination pages together.
6. Write your answers in the space provided. Use the back of pages if necessary.
7. The spare page(s) are for your own use. They can be detached from the exam. Their content will not be graded.
8. Peeking at your neighbours' work will result in expulsion from the exam.

Problem	1	2	3	4	5	6	Bonus			
Maximum	2	3	26	8	5	6	2			
Grade								**Total**		/ 50

Problem 1 [2 points]

Consider the following predicate definition in Prolog:

```
guess(Data, Result) :-
  append(X, Y, Data),
  length(X, LX),
  length(Y, LY),
  abs(LX - LY) =< 1,
  append(Y, X, Result).
```

Now, consider the execution of the query

```
?- guess([a, b, c, d, e, f, g], Z).
```

when we repeatedly enter the semicolon. What result will we get?

A	Z = [e, f, g, a, b, c, d] ; Z = [d, e, f, g, a, b, c] ; false.	B	Z = [d, e, f, g, a, b, c] ; Z = [e, f, g, a, b, c, d] ; false.
C	Z = [g, f, e, d, c, b, a] ; false.	D	Z = [a, b, c, d, e, f, g] ; Z = [b, c, d, e, f, g, a] ; false.

Answer: B

Problem 2 [3 points]. Write a Scheme function that returns the number of ones (1) in a given simple list of numbers. Please write the function using only elementary constructs such as car, cdr, etc.

```
(define (count-ones lis)
     (cond
          ((null? lis) 0)
          ((eq? 1 (car lis)) (+ 1 (count-ones (cdr lis))))
          (else (count-ones (cdr lis)))
     )
)
```

Problem 3 [26 points] Consider the following grammar, where **NP** is the start symbol, non-terminal symbols are indicated in bold type and ε is the empty string:

NP → **Name** | **Art Adjs N PP**
Adjs → **Adj** | **Adjs** and **Adj** | **Adj Comma Adjs** | ε
PP → **Rel Verb NP Colon** | ε
Name → John | Amy
Art → The | A
Adj → nice | ugly | expensive | sweet | green | sad

Comma → **,**

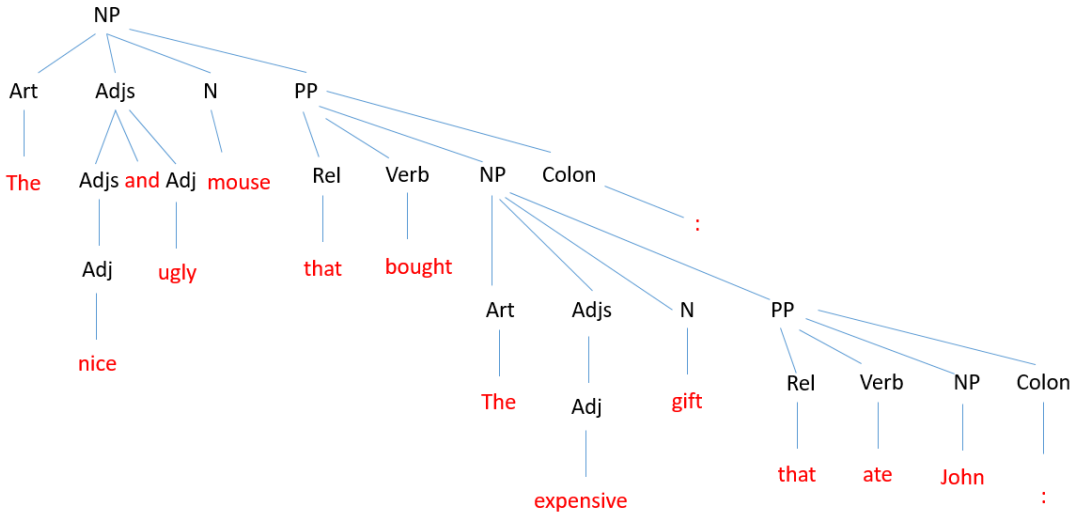
Colon → **:** | ε
N → mouse | cat | gift
Rel → that | who
Verb → ate | bought

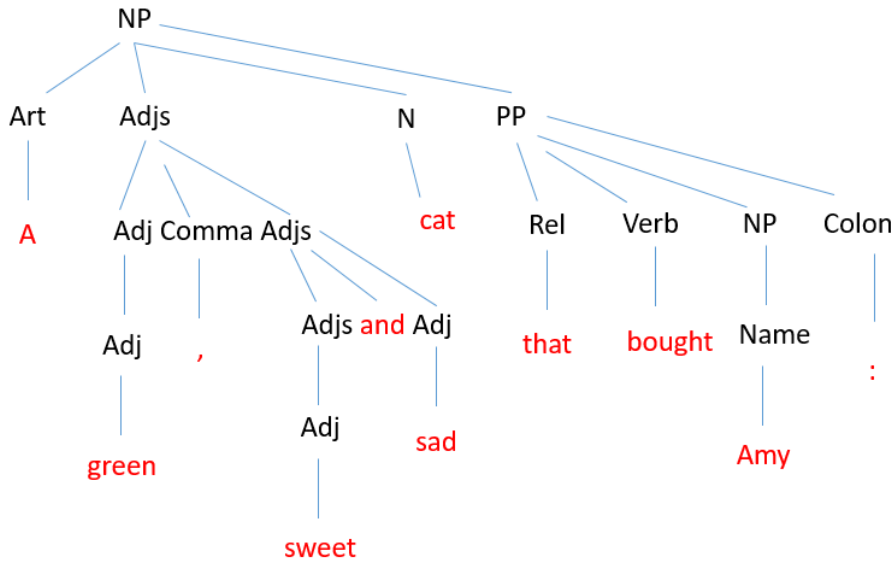
3.1 **[6 points]** Generate two sentences (actually Noun Phrases) that have correct English grammar, though not necessarily meaningful semantics. These sentences should contain at least 8 words each (excluding punctuation) and should be represented by significantly different parse trees.

(1) The nice and ugly mouse that bought The expensive gift that ate John

(2) A green, sweet and sad cat that bought Amy

3.2 **[6 points]** Draw the parse trees for sentences (1) and (2) of subquestion 1.





3.3 **[4 + 4 = 8 points]** Show the contents of *First* and *Follow* sets of **Adjs** and **Verb.**

NP → **Name** | **Art Adjs N PP**
Adjs → **Adj** | **Adjs** and **Adj** | **Adj Comma Adjs** | ε
PP → **Rel Verb NP Colon** | ε
Name → John | Amy
Art → The | A
Adj → nice | ugly | expensive | sweet | green | sad

Comma→ **,**

Colon → **:** | ε
N → mouse | cat | gift
Rel → that | who
Verb → ate | bought

	First	Follow
Adjs	Adj | ε nice, ugly, expensive, sweet, green, sad, ε	N | and mouse, cat, gift, and
Verb	ate, bought	Name | Art John, Amy, The, A

3.4 **[3 points]** The Adjs rule is left-recursive. Rewrite that rule in a way that eliminates the left recursion.

Original rule
Adjs → **Adj** | **Adjs** and **Adj** | **Adj Comma Adjs** | ε

Transformed rule
Adjs → **Adj Adjs'** | **Adj Comma Adjs Adjs'** | ε
Adjs' → and **Adj Adjs'**| ε

3.5 **[3 points]** Left-factor the Adjs rule that you obtained in subquestion 5.

Adjs → **Adj Adjs''** | ε
Adjs' → and **Adj Adjs'**| ε
Adjs'' → **Adjs' | Comma Adjs Adjs'**

Problem 4 [8 points]

Consider the following statement (**x**, **y**, **p** are variables of the same numeric type):

```
if (x < y) {p = x; x = y; y = p;}
```

3.1 **[4 points]** The correct precondition-postcondition pair for this statement is

	precondition	postcondition
A	`true`	`x ≥ y`
B	`x < y`	`x ≥ y`
C	`true`	`x > y`
D	`x ≤ y`	`x > y`

Answer: A

3.2 [4 points] Prove that given the postcondition you selected in problem 3.1, it is possible to arrive at your selected precondition.

Let us compute the precondition for the statement block S starting from the selected post-condition

if (x < y) {

{y >= x} p = x; {y >= p}
{y >= p} x = y; {x >= p}
{x >= p} y = p; {x >= y}

}

{x >= y} ← the selected post-condition

Now we need to prove two things with this precondition {y >= x}:

- That we have S with precondition {B and P} = {x < y and true} and postcondition Q = {x >= y}
    - {B and P} reduces to {x < y}. By the rule of consequence we have x < y => x <= y, or, put differently, x < y => y >= x. Notice that y >= x is the precondition for the S block derived from the selected post-condition.
- That (not B and P) => Q

  (not (x < y) and true) => x >= y
  x >= y and true => x >= y
  x >= y => x >= y
  Hence the precondition for the if statement is {true} and the post-condition is {x >= y}.

Problem 5 [5 points]

Write True (T) or False (F) in front of the following statements. Justify your rationale for the False (F) statements.

_____ Axiomatic semantics is the most rigorous approach to describe the semantics of a language since it is based on a set of mathematical functions associated with the language objects and constructs.

FALSE; denotational semantics is concerned with capturing the semantics of a language via a rigorous definition of mathematical functions associated with the language objects and constructs. Axiomatic semantics is based on predicate logic.

_____ Multiple derivations in an unambiguous grammar could lead to the same parse tree.

TRUE; a leftmost and rightmost derivations of the same sentence will lead to the same parse tree in an unambiguous grammar.

_____ Simula 67 introduced the central concepts of object-oriented programming, namely classes and encapsulation.

TRUE; Simula 67 was a predecessor of object-oriented languages such as SmallTalk and C++.

_____ A top-down parser can process a wider number of languages than a bottom-up parser can.

FALSE; bottom-up parsers can process a wider number of languages than top-down parsers given that they don't suffer from issues like left recursion or left factoring.

_____ Prolog has been successfully used to develop expert systems.

TRUE; this is one of the two major Prolog application areas discussed in class. The other one is relational database management systems.

Problem 6 [6 points]

List three characteristics that help make a language readable. Please illustrate your response with examples.

We discussed four aspects in class that contribute to the readability of a programming language, namely:

1. Simplicity:
   - Keeping the number of basic constructs (primitives, etc.) rather small
   - Having minimal feature multiplicity, for example trying to avoid scenarios such as count = count +1, count +=1, count++ and ++count.
   - Minimal operator overloading, as the possibility to overload any operator could lead to misleading definitions, for example defining the sum of two arrays a + b as the total sum of their coordinates instead of the pairwise sum of their coordinates.
2. Orthogonality:
   - Giving a fair chance to most primitive constructs to be applied to more complex constructs, e.g., pointers, structs and arrays. This means reducing the number of special cases in which an operator does not apply to a primitive type.
   - For instance, in C an array can contain any data type but void. A function can return a struct (record) but not an array.
3. Data types: Introducing data types such as Boolean that clarify the meaning of some statements, e.g., timeOut = true vs. timeOut = 1.
4. Syntax design:
   - Disallowing variables being named as language keywords
   - Having the language keywords clearly represent their meaning, e.g., class, while, for, package, etc.
   - Using more specific compound statements (end if, end for) that just }.

Bonus [2 points]:

Mention two large companies and at least one programming language that each has developed.

Google: Go, Dart
IBM: Fortran, PL/I
Microsoft: C#, VB.NET, F#, VBA
Sun Microsystems/Oracle: Java
Yahoo: Hadoop
Ericsson: Erlang

Spare page 1

Spare page 2