MEMORANDUM

TO: All **P**roud **C**ompilers **S**tudents in CST8152
FROM: Prudent teacher (Sv. Ranev)
DATE: 4 April 2016
SUBJECT: Syntax-Error-Free Parsing of the Compilers Final Exam

HINT: COGITO ERGO SUM

Section 010 **Test date:** 29 April 2016 **Test time**: 8:00-10:30 **Test room:** P303

The Final Exam is a comprehensive and relatively easy test on all of the material covered in the course. The Final exam consists of 30 questions. They are broken down into thee groups – Guess (True-False), Multiple-choice, and Do-it-Yourself questions. The questions will have different weight. At the end of test there will be three bonus questions. The test will count for 30% of your final marks.

A prudent PCS will read carefully the memo describing the Mid-term test and will refresh their memory and skills. When done, a prudent PCS will concentrate on the second part of the course material and will pay special attention to the following additional pages in chapters 2, 4 and [7fe]:

Second edition: 64 - 68, 85 - 91, 72 -73, 195 -196, 199 - 202, 213 - 231, 233 - 234

First edition: 40 - 48, 60 - 62, 160 - 198, 215 - 216, 257 - 261, 429 - 440

A prudent PCS student will establish very close relations with:

symbol tables: purpose, functions, and implementation details; ***syntax analysis***: the role of the parser, syntax errors and error recovery, parse tree and derivations, top-down and bottom-up parsers, predictive parsers, grammar transformations for predictive parsers, first and follow sets, non-recursive predictive parsers, parsing tables and parsing by non-recursive predictive parsers.

A prudent PCS student will have serviceable knowledge of how to:

build parse threes and derivations; eliminate a left recursion in a grammar production; apply left factoring transformations; build the first and follow sets for a grammar production; build a parsing table for a predictive parser; describe the moves made by predictive parser; construct simple parse trees.

A prudent PCS student will be familiar with all the figures in the specified pages.

A prudent PCS student will peruse their lecture notes and the PLATYPUS language grammar.

IMAGINE ALL THE STUDENTS ...

Good Luck -> PSD **|** ε
P -> Prepare
S -> Study
D->DoOtherThings

EXAMPLES OF THE TEST QUESTIONS:

1.- What should you do to pass the Final Test?

a. Study
b. Eat
c. Drink
d. Breathe
e. Sleep
f. Relax
g. Enjoy
h. all of the above

The correct answer is h.

2.- Predictive parsing is a form of top down parsing which allows backtracking to occur.

a. True b. False

The correct answer is b [p 219][old: p183].

3.- Given the grammar:

S -> iEtS | iEtSeS | a
E -> b

Write the equivalent left-factored grammar.

The correct answer is on page 215 [179].

4. Show that the following grammar is ambiguous using the given sentence as an example.
Sentence: *if 1 then if 0 then a else b*
Grammar:

Stmt -> if Cond then Stmt | if Cond then Stmt else Stmt | Other
Cond -> 1 | 0
Other -> a | b

The correct answer is on page 174 [210] and in the notes.

5. Given the grammar below, apply different transformations in order to obtain a grammar that can be parsed by a recursive-descent parser that needs no backtracking.

Grammar:

A -> v+a | A=F
F -> v | (E)
E -> Fs | s

This is one more example of how to transform a grammar, build the FIRST, the FOLLOW, and the predictive parsing table for the transformed grammar.

Given the grammar below, apply different transformations in order to obtain a grammar that can be parsed by a recursive-descent predictive parser. Build the FIRST and FOLLOW sets for the transformed grammar. Build the predictive parsing table for the transformed grammar.
Lower-case letters denote terminals; Upper-case letters denote nonterminals.

Grammar:

A -> v+a | A=F
F -> v | (E)
E -> Fs | s

Answer:

A -> v+a A’ (A = A α | β, where α = **=F**, β = **v+a**)
A’ -> =FA’ | ε
F -> v | (E)
E -> Fs | s

The FIRST set for the transformed grammar above is:

FIRST(A) = {FIRST(v+aA’)} = {FIRST(v)} = {v}
FIRST(A’) = {FIRST(=FA’), **ε**} = {FIRST(=), **ε**} = {=, **ε**}
FIRST(F) = {FIRST(v) , FIRST((E))} = {v, (}
FIRST(E) = {FIRST(F), FIRST(s)} = {v, (, s }

The FOLLOW set for the transformed grammar above is:

FOLLOW(A) = {$}
FOLLOW(A’) = {FOLLOW(A) , FOLLOW(A’)} = {$,$} = {$}
FOLLOW(F) = {FIRST(A’), FIRST(s)} = {=, FOLLOW(A’), s} = {=, $, s}
FOLLOW(E) = { FIRST( )) } = {) }

The predictive parsing table for the grammar above is:

Nonterminals	Input Tokens							
	v	**=**	**(**	**)**	**a**	**s**	**+**	**$**
A	v+aA’							
A’		=FA’						**ε**
F	v		(E)					
E	Fs		Fs			s		

Using the table, try to parse the following sentence: **v+a= (vs)**
The following left-most derivation proves that the sentence is syntactically correct and can be generated by the grammar.
A$ => v+aA’$ => v+a=FA’$ => v+a = (E)A’$ => v+a= (Fs)A’$ => v+a=(vs) **ε**$ => **v+a = (vs)$**

Enjoy and do not forget that

"The best advice given to the students is: 'Find out what you like doing best and get someone to pay you for doing it'." Katherine Withehorn

Also, do not forget what you have learned in this course:

"Reeling and Writhing, of course, to begin with," the Mock Turtle replied; "and then different branches of Arithmetic – Ambition, Distraction, Uglification, and Derision."

"I never heard of 'Uglification,'" Alice ventured to say. "What is it?"
Alice's Adventure in Wonderland, Lewis Carroll (Charles Lutwidge Dodgson)

sVillain Raven lifted up both its paws in surprise. "Never heard of uglifying!" he exclaimed. "You ought to be ashamed of yourself for asking such a simple question. You know what to ***compile*** *is, I suppose?"*

*"Yes," said PCS doubtfully: "it means—to—make—anything—**a-bit-wise**."*
"Well, then," sVillain Raven went on, "if you don't know what to uglify is, you are a Platypus."
Students' Adventure in Compilerland

Professor: Svillen Ranev
CST 8152 – Compilers
04/04/2016