Exam of Theoretical Computer Science. Year 2010–2011. A. Pettorossi.

1. **Decidability and Partial Recursive Functions**: • r.e. sets and recursive sets. • Semidecidability of the Halting Problem of the Turing Machine. • Partial Recursive Functions. (AP: all proofs are optional.)

2. **Predicate Calculus and Logic Programming**: • Syntax and semantics: Classical Presentation. • Operational and Denotational Semantics of Definite Logic Programs. (PC&LP: Sections: 1, 2.1, 3, 4, 8, 9.3, 9.4. All proofs are optional.)

3. **Well-founded Recursion Theorem** (SPL: Section 3.1.6. No proof)

4. **Language IMP**: • Operational Semantics. • Denotational Semantics. • Axiomatic Semantics. • Rule induction. • Soundness of Axiomatic Semantics. • Weakest Preconditions, Expressiveness Theorem, and Relative Completeness Theorem. (SPL: No proofs)

5. **Language REC**: • Operational Semantics and Denotational Semantics in call-by-value and in call-by-name. • Equivalence of semantics. • The Factorial example in the two languages. (SPL: No proofs)

6. **Complete partial orders**. • Products, function space, curry, apply, lifting, sum, and Cond. • Metalanguage for denotational semantics. • Fixpoint of continuous functions and Kleene Theorem. • Scott fixpoint induction, Park induction, and Bekić Theorem. (SPL: Basic definitions and properties. Proof of Kleene's Theorem. No proofs of the other theorems.)

7. **Higher order languages**: • The Eager language, the Lazy1 language, and Lazy2 language: Operational Semantics and Denotational Semantics. • The Factorial Example in the three languages. • The β rule and the η rule in Eager, Lazy1, and Lazy2. (SPL: Basic concepts only. No proofs)
Optional: Fixpoint operators in Eager, Lazy1, and Lazy2. *Optional*: Adequacy and full-abstraction.

8. **Non-determinism and Concurrency**. • Owicki-Gries rules. • Milner's CCS. • Axiomatization of finite state processes. • Hennessy-Milner logic. • Modal μ-calculus. • Local model checking. • Correctness proof of the Alternating Bit Protocol. (SPL: No proofs).
Optional: Dijkstra's guarded commands. *Optional*: Bisimulation equivalence and Bisimulation congruence for CCS.

9. **Projects**: (A1) Define a higher order lazy language, call it EL, which is an extension of the Lazy language and write a Prolog program for the operational semantics of EL.
(A2) Write a simple local model checker in Prolog and use it for proving the correctness of a mutual exclusion protocol or a cache coherence protocol.

References.
- AP: Pettorossi, A.: *Elements of Computability, Decidability, and Complexity.* Third Edition. Aracne (2009).
- PC&LP: A. Pettorossi and M. Proietti: *Predicate Calculus and Logic Programming.* Second Edition. Aracne 2005.
- SPL: Pettorossi, A.: *Semantics of Programming Languages.* Aracne (2010).