Exam

Programming Languages and Compilers

SPO EXAM, spring term 2019

The exam is oral and without preparation time. Each student will have a ca. 20-minute session. The student will draw a question and give a short presentation about the subject in the first part of the session, max 10 minutes. In the second part of the session there will be a discussion between the student, lecturer and examiner.

Many subjects are rather broad and for each subject there is a listing of sub-topics you could discuss, but you do not have to cover all of them in your presentation. It is your responsibility to select parts of the subject that you consider to be most relevant.

For each question there will be a set of slides available that you can choose to use for your presentation and subsequent discussion – note you do not need to use all the available slides.

You will not be allowed to use your own slides.

You are allowed to bring your own notes.

Note: You may choose to have your examination in Danish or English – just let us know which language at the beginning of the session.

Syllabus

Fischer et. al., chapter 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13

Sebesta, chapter 1, 3.1-3.4, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14

(Note that Sebesta chapter 2, 3.5, 13, 15, 16 and Fischer et. al. chapter 14 are not part of the syllabus)

Articles and web references used during the course are not considered part of the syllabus, but may serve you as background material for further elaboration.

Questions

The student should be able to describe techniques and concepts in programming language design and compiler construction using correct terminology and notation to describe implementations of programming languages. The student should be able to describe how implementation techniques influence programming language design. The student should be able to describe, analyse and implement programming languages and explain the compiler phases and connections between them.

1. Language Design and Control Structures
    a. Language Design Criteria

   b. Evaluation of expressions
   c. Explicit sequence control vs. structured sequence control
   d. Loop constructs
   e. Subprogram
   f. Parameter mechanisms
   g. Slides
2. Structure of the compiler
   a. Describe the phases of the compiler and give an overall description of what the purpose of each phase is and how the phases interface
   b. Single pass vs. multi pass compiler
      i. Issues in language design
      ii. Issues in code generation
   c. Slides
3. Lexical analysis
   a. Describe the role of the lexical analysis phase
   b. Describe how a scanner can be implemented by hand or auto-generated
   c. Describe regular expressions and finite automata
   d. Slides
4. Parsing
   a. Describe the purpose of the parser
   b. Discuss top down vs. bottom up parsing
   c. Explain necessary conditions for construction of recursive decent parsers
   d. Discuss the construction of an RD parser from a grammar
   e. Discuss bottom Up/LR parsing
   f. Slides
5. Semantic Analysis
   a. Describe the purpose of the Semantic analysis phase
   b. Discuss Identification and type checking
   c. Discuss scopes/block structure and implication for implementation of identification tables/symbol tables
   d. Discuss Implementation of semantic analysis
   e. Slides
6. Run-time organization
   a. Data representation (direct vs. indirect)
   b. Storage allocation strategies: static vs. stack dynamic
   c. Activation records (sometimes called frames)
   d. Routines and Parameter passing
   e. Slides
7. Heap allocation and Garbage Collection
   a. Why may we need heap allocation?
   b. Garbage collection strategies (Types of GCs)
   c. Slides
8. Code Generation
   a. Describe the purpose of the code generator
   b. Discuss Intermediate representations
   c. Describe issues in code generation
   d. Code templates and implementations
   e. Back patching
   f. Implementation of functions/procedures/methods
   g. Register Allocation and Code Scheduling
   h. Slides