

- Adapted from [Revision Guide \(revised 2017\)](#)

I. Introduction and motivation

- design, methods, paradigms; Foundations; Standardisation.
- [y2006p6q7 \(a\)](#)
 - motivating application domains, abstract machines, theoretical understanding
- [y2012p3q6 \(a\)](#)
 - Execution models (abstract machines)
 - Storage allocation and deallocation
- [y2015p3q5 \(a, b\)](#)
 - Programming-language concepts, innovations, influences
- [y2007p6q7 \(b,c\)](#)
 - Parameter passing, value, reference, value/result, name
 - Aliasing

II. FORTRAN: A simple procedural language

- FORTRAN 77; Data types; Control structures; Syntax; Storage; Aliasing; Parameters.
- [y2006p6q7 \(b\)](#)
 - Types, advantages and disadvantage
- [y2007p5q7 \(a\)](#)
 - Execution model (or abstract machine)
 - Compilation
- 2009 Paper 3 Question 2 (a)
- 2010 Paper 3 Question 5 (a)

III. LISP: Functions, recursion, and lists

- LISP; Programming-Language phrases; S-expressions; quote; Abstract machine; Recursion; Programs as data; Reflection
- [y2006p6q7 \(c\)](#)
 - Static and Dynamic scope
- [y2007p6q7 \(a\)](#)
 - Execution model (or abstract machine)
 - Compilation
- 2009 Paper 3 Question 2 (a)
- 2014 Paper 3 Question 6 (a)
- [y2007p5q7 \(b\)](#)
 - Garbage collection
- 2008 Paper 6 Question 7 (a)

- 2011 Paper 3 Question 6 (a (ii))

IV. Block-structured procedural languages – Algol and Pascal

- Block structure; Algol 60; Recursion; Stack; Type system; Algol 68; BNF syntax; Heap; Garbage collection; Quasi-strong typing;
- [y2006p6q7 \(b\)](#)
 - Types, advantages and disadvantage
- 2008 Paper 5 Question 7 (a)
- 2010 Paper 3 Question 5 (a)
- [y2012p3q6 \(c\)](#)
- [y2007p5q7 \(c\)](#)
 - Algol 60 primitive static type system, Parameter-passing
- 2013 Paper 3 Question 6 (b)
- 2008 Paper 6 Question 7 (b)
- 2009 Paper 3 Question 2 (c)
- 2011 Paper 3 Question 6 (a (i))
- 2013 Paper 3 Question 6 (a (i))
- [y2015p3q5 \(c\)](#)
 - Pascal variant records vs ML vs subclass

V. Object-oriented languages – SIMULA and Smalltalk

- Objects in SML; Subtyping vs. inheritance; SIMULA; Classes, objects and activation records; Subclasses and inheritance; Smalltalk; Dynabook; Syntax; Abstraction; Messages; Methods; Instance variables; Interfaces as types; Subtyping.
- [y2006p6q7 \(d\)](#)
 - Dynamic lookup; Abstraction; Subtyping; Inheritance
- 2008 Paper 5 Question 7 (c)
- 2011 Paper 3 Question 6 (a)
- 2013 Paper 3 Question 6 (a (ii))
- [y2007p6q7 \(d\)](#)
 - SIMULA, Type checking and subtyping
- 2010 Paper 3 Question 5 (b)
- [y2012p3q6 \(f\)](#)
 - Abstraction, `private` weakened by pointer / reflection

VI. Types in programming languages

- Type checking in SML; Type equality; Type declarations; Type inference; Type inference in SML; let-polymorphism; Polymorphic exceptions.
- 2008 Paper 5 Question 7 (b)
- 2010 Paper 3 Question 5 (c)
- [y2012p3q6 \(b\)](#)
- [y2015p3q5 \(d,e\)](#)

- static vs dynamic scoping, early LISP
 - static vs dynamic type checking
 - type-safe and counterexample
- 2013 Paper 3 Question 6 (c)
- [y2015p3q5 \(e, f\)](#)
 - downcast
- 2009 Paper 3 Question 2 (b)
- 2011 Paper 3 Question 6 (b)
- [y2012p3q6 \(e\)](#)
 - Type safety and counterexample
 - Polymorphism in ML
- 2014 Paper 3 Question 6 (c)
- 2016 Paper 3 Question 5 (d, e)

VII. Scripting Languages – JavaScript

- Scripting vs. dynamic typing; JavaScript; Prototypal inheritance; Browser integration. Students might consider the following questions:
- “Scripting languages and dynamically typed languages are identical; discuss”
- “Discuss the notion of ‘class’ in relation to JavaScript”

VIII. Data abstraction and modularity – SML Modules

- Concrete and opaque signatures; Signature inclusion; Signature matching; Subtyping; Information hiding; Functors.
- [y2007p5q7 \(d\)](#)
 - SML module system, Signatures; Structures; ADT of stacks
 - Functional / Imperative
- 2010 Paper 3 Question 5 (d)
- 2013 Paper 3 Question 6 (d)
- 2009 Paper 3 Question 2 (d)
- 2011 Paper 3 Question 6 (c)
- 2014 Paper 3 Question 6 (d)

IX. Languages for concurrency and parallelism.

- Theoretical models; Threads, shared memory, message passing; Distributed memory, multi-core, cloud computing; Programming-language support for parallelism and distribution. Internal and external iteration.
- 2014 Paper 3 Question 6 (b)

X. Functional-style programming meets object-orientation.

- Scala and Java 8; Procedural programming; Declarative programming; Mutable state; Blocks; Functions; Parameter passing; Classes and objects; abstract classes; traits; case classes; Pattern matching; Functions as objects;

- [No longer explicitly lectured:] Type parameter bounds; View bounds; Implicit parameters; Implicit conversions; Mixin-class composition.
- 2008 Paper 6 Question 7 (c)
- 2010 Paper 3 Question 5 (e)
- [y2012p3q6 \(d\)](#)
 - Generic types and methods; Variance annotations
- 2009 Paper 3 Question 2 (e)
- 2011 Paper 3 Question 6 (d)
- 2013 Paper 3 Question 6 (e)

XI. Miscellaneous concepts Keywords:

- Monads, GADTs, Reified continuations, Dependent typing.
- 2016 Paper 3 Question 5 (a, b, c)