Lecture 2:

Lambda calculus  
  
1) What is a function in Lambda calculus?  
  
2) Present the Lambda notation.  
  
3) Present the syntax of Lambda calculus.  
  
4) Describe the concepts of the operation substitution, the Alpha conversion, and the Beta reduction.  
  
5) How to evaluate a Lambda expression?  
  
6) How can we represent and compute with Boolean values in Lambda calculus?  
  
6) How can we represent and compute integer numbers in Lambda calculus?  
  
7) What are combinators? Present the essential combinators.  
  
8) How can we model the recursion in Lambda calculus?  
  
9) Present some important properties of Lambda calculus.

Lecture 3 - 4:

Functional programming languages  
  
1) What are the historical origins of functional languages?  
  
2) Explain the concept of value in functional languages. Present the possible types of values.  
  
3) Present the data structures used in functional languages.  
  
4) Present a product data structure in functional languages.  
  
5) Describe in what ways the lambda abstraction is used in functional languages.  
  
6) Present the concept of a function in OCaml.  
  
7) Describe the Curry form of a function. What are the benefits of using a function in the Curry form?  
  
8) Describe the ways to define the binding between a name and a value in an OCaml block.  
  
9) Present the concepts of a scope and a lifetime of a variable.  
  
10) Describe the concept of recursion in functional languages.  
  
11) Present the parametric polymorphism.  
  
12) Present a higher-order function. How can we use them?  
  
13) Describe pattern matching in OCaml.  
  
14) How to define a parameterized (polymorphic) type in OCaml?  
  
15) Describe a union type in OCaml.

Lecture 5 – 6:

Imperative programming languages  
  
1) Present the differences between the functional and imperative models of computation.  
  
2) What was the idea of the structured programming developed around 1970?  
  
3) Describe the concept of a variable in imperative programming languages (abbr. IPL).  
  
4) Describe the concepts of imperative languages used for expressing sequential control.  
  
5) Present the concepts of a procedure and a function in IPL.  
  
6) Describe the most important parameter passing methods.  
  
7) Present the record data structure and its use in IPL.  
  
8) Present the concept of a pointer in IPL.  
  
9) Present the array data structure and its role in IPL.  
  
10) Describe the set data structure and its use in IPL.  
  
11) Describe the dictionary data structure (associative array) and its role in IPL.

Lecture 7:

Compilers and interpreters  
  
1) Present the structure of a compiler.  
  
2) Describe the process of line reconstruction.  
  
3) What is lexical analysis?  
  
4) What is the grammar of a programming language?  
  
5) Describe a parser.  
  
6) Describe the tasks of the semantic analysis.  
  
7) Describe the role of intermediate code.  
  
8) Present the approaches to intermediate code optimization.  
  
9) Describe the functions of the back-end.

Lecture 8 – 9:

Object-oriented programming languages  
  
1) Describe the object-oriented model of programming languages.  
  
2) Describe the definition of a class and the creation of instances.  
  
3) Describe the aggregation abstraction and its implementation in object-oriented languages.  
  
4) Present the specialization abstraction and its implementation in object-oriented languages.  
  
5) Describe the concept of method overriding.  
  
6) Describe the concept of multiple inheritances and the problems related to multiple inheritance.  
  
7) What is an abstract data type? Give an example.  
  
8) Present the concepts of subtyping and substitutivity.  
  
9) Present the dynamic binding and the subsumption polymorphism.  
  
10) Describe the implementation of classes and objects.  
  
11) Present the concept of an abstract class.  
  
12) What is genericity? Describe the concept of a parameterized class.

Lecture 10 – 11:

Memory management  
  
1) Describe the concept of \*binding time\*.  
  
2) Describe the lifetime of objects and bindings.  
  
3) Present the general approaches to the storage allocations in programming languages.  
  
4) Present the static allocation of memory in programs. What is a static activation record?  
  
5) Describe the stack-based allocation of the space for subroutines. Detail the role of activation records.  
  
6) Present the heap-based allocation of the space for objects. What problems appear with the heap-based allocation?  
  
7) What strategies do we use to manage space in a heap?  
  
8) Describe explicit memory management in programming languages.  
  
9) Describe automatic memory management in programming languages. Why garbage collection?  
  
10) Discuss the trade-off between automatic garbage collection and explicit memory management.  
  
11) Describe garbage collectors based on reference counts.  
allocation  
12) Describe the mark-and-sweep mechanism for garbage collection. Present the possible improvements of mark-and-sweep.

Modular programming languages  
  
1) Describe the concept of a module in programming languages.  
  
2) Present the use of modules as compilation units.  
  
3) Present the main characteristics of modules in C and C++ programming languages.  
  
4) Describe the module system in the programming language Java.  
  
5) Describe the concept of an abstract data type and its relations to modules.  
  
6) Describe the concepts of an interface and an implementation of modules in Ocaml.  
  
7) Present the constructs of the module language in Ocaml.  
  
8) How to achieve information hiding in modules? How to define multiple views of a module in Ocaml?  
  
9) Describe parameterized modules (functors). Give an example of a parameterized module.