Give 4 different programming paradigms and briefly explain the underlying principle for each paradigm. [6]

Give 4 different drivers of the evolution of programming languages and briefly explain each of these driving motivations. [4]

In order to execute a high-level program we must first convert it into machine code so that the CPU can ‘understand’ it. Briefly explain the difference between compilation and interpretation. How are Java programs executed? [6]

Imperative: Statements change a programs state (closest to "memory abstraction" of CPU)
Declarative: Programs say what to do, rather than how to do it
Data-Oriented: Programs work with data through manipulating and searching relations (tables). Tables have things in common that can be linked together to get more information
Scripting: Designed to automate frequently used tasks that involve calling or passing commands to external programs. These languages have lots of libraries to make things easier to do

Productivity: Speed up software development process, reduce times and costs, support fast user interface development. This lead to the development of rapid applications development (RAD) languages and scripting languages
Reliability: To try and reduce the number of errors caused during the execution of the program, this includes things such as type checking and exception handling.
Security: Scripting languages used for webpages can, when run on machines, enable malicious programmers to breach your security.
Execution: Different programming languages will work better for multi-threading and multi-core processing, allowing parallel computing

Compilation is where the entire program is converted into a form that can be executed by the processor before the program is run, whilst interpretation is where the program is converted into an executable form whilst it is being run.
Java, however, is compiled into bytecode before run time, this bytecode is then interpreted at runtime.

Outline a more refined view of (the phases of) compilation. [8]

Outline how you would develop a Prolog program that given some facts about who is the mother or father of whom (in some collection of individuals), computes who is the grandmother, grandfather or descendant of whom. [8]

What are the syntax and the semantics of a programming language? Give 2 reasons why we should strive for a formal semantics for a programming language. [6]

- Lexical analysis: converts the program into basic syntactic components and converts it into a token stream.
- Syntax analysis: converts the token stream into a parse tree.
- Translation phase: converts the parse tree into a linear sequence of intermediate code.
- Code generation: converts the intermediate code into assembly and then machine code.

```
grandma(X,Y) :- mother(X,Z), parents(_,Z,Y).
grandma(X,Y) :- mother(X,Z), father(Z,Y).
grandpa(X,Y) :- father(X,Z), parents(_,Z,Y).
grandpa(X,Y) :- father(X,Z), father(Z,Y).
descend(X,Y) :- mother(X,Y)
descend(X,Y) :- father(X,Y)
descend(X,Y) :- descend(X,Z) , descend(Z,Y)
```

Syntax: the rules the govern what makes a program 'legitimately written'
Semantics: the rules which govern what a program 'means'
Formal semantics are needed in a programming language so that the programmer is left in no doubt as to what the program will do when it is executed. Without formal semantics we can no longer prove that a program will do what it is intended to do or even runs the same on different machines (with no semantics, processors may interpret the commands differently).

Define carefully what a regular expression is and explain how a regular expression is used to denote a set of strings. [10]

A **regular expression** over some **alphabet** Σ (finite set of symbols)

- Any a $\in \Sigma$ is a regular expression. A regular expression is any letter from the alphabet
- ø(empty set) and ϵ (empty string) are regular expressions
- If ω and ω' are regular expressions then so are: