What is computer software?

1. Briefly (and, necessarily, without being too precise) explain the difference between an algorithm and a program. Suppose we have an algorithm and wish to implement it in Python. How many different implementations are there of this algorithm? [5]

Solution:

An algorithm is a sequence of precise instructions that can be applied to specific data items. A program is the implantation of the algorithm in a form that can be executed by a computer, or at least compiled to a form that can be executed by a computer. There are many different implementations of an algorithm as a program.

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

Solution:

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

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

Solution:

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

4. Give 3 general properties any programming language should have. [3]

Solution:

- Be easy to use, with its programs easy to read, write and understand
- Support abstraction so that adding new features and concepts should be possible
- Support testing, debugging and program verification
- Be inexpensive to use, in terms of execution time, memory usage and maintenance costs

5. Explain very briefly how a Prolog program computes. [4]

Solution:

A Prolog program consists of a list of facts (atoms) and rules that can be applied to the facts. It then takes queries about the facts which it can answer using the atoms and the rules.

6. 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]

Solution:

```
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)
```

7. What is the fundamental principle of the research area known as ubiquitous computing? [2]

Solution:

The integration of computers and software into everyday objects and activities so that we can control remote aspects of our lives, mostly through RFID.

8. Give two illustrations of principles of Computational Thinking in the context of software. [2]

Solution:

Green Computing:An area of computer science involving energy conservation within the world of information technology. This involves writing the main unit of resource is energy expended.
Parallel processing: Certain kinds of problems can be conveniently represented as multiple communicating threads which help to structure code in a more modular manner, e.g., by modelling user interface components as separate threads.

9. 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]

Solution:

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)