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 implementation 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 application 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)

10. 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]

Solution:

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 while it is run.
Java, however, is compiled into bytecode before run time, this bytecode is then interpreted at runtime

11. Give an advantage of compilation over interpretation, and of interpretation over compilation [2]

Solution:

Compiled programs have faster execution
Interpretation is faster in a development environment where a program is constantly being changed

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

Solution:

- 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

13. In compilation, over 50% of the time taken can be spent on lexical analysis; that is, character handling. In moving from a program as a string of symbols to a token stream, which algebraic construction is usually used to define tokens? [2]

Solution:

Regular Expressions

14. Define carefully what a regular expression is and explain how a regular expression is used to denote a set of strings [14]

Solution:

A regular expression over some alphabet Σ is defined as follows:

- Any $a \in \Sigma$ is a regular expression
- $\emptyset$ and ϵ are special regular expressions

- If ω and ω' are regular expressions, then so are
  - $(\omega\omega')$
  - $(\omega|\omega')$
  - (ω^*)

Every regular expression denotes a set of strings over Σ

- $a, \emptyset$ and ϵ denote the sets of strings $\{a\},$ and $\{\epsilon\}$ respectively
- If ω and ω' denote the sets of strings R and S then:
  - $(\omega\omega')$ denotes $\{xy : x \in R, y \in S\}$
  - $(\omega|\omega')$ denotes $R \cup S$
  - (ω^*) denotes $\{x_1x_2 \dots x_n : n \geq 0, x_1, x_2, \dots, x_n \in R\}$

15. Give a regular expression that denotes the set of strings over the alphabet $\{a, b, c\}$ consisting of strings with the property that there is always at most one c [5]

Solution:

$$((a|b)^*|c)(a|b)^*$$

16. Give a regular expression that denotes the set of strings over the alphabet $\{a, b\}$ consisting of strings with the property that a must always be followed by bb [7]

Solution:

$(b|abb)^*$

17. Define carefully a finite state machine and explain how one is used to accept a set of strings [8]

Solution:

A finite state machine is defined as:

$$M = (\Sigma, Q, \delta : Q \times \Sigma \to Q, q_0 \in Q, F)$$

where

- Σ is some finite alphabet
- Q is some finite set of states with initial state q_0 and set of final states $F \subseteq Q$
- $\delta : Q \times \Sigma \to Q$ is the transition function

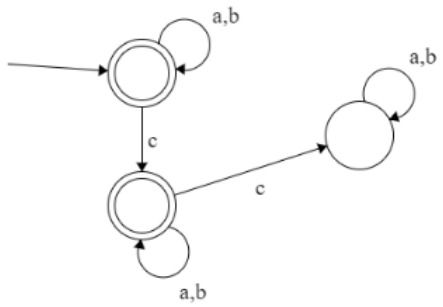
On input any string $a_1a_2...a_n$ over Σ (where $n \geq 0$, and so we may input the empty string if we wish), M yields a sequence of states $q_0, q_1, q_2, ..., q_n$ via:

$$q_1 = \delta(q_0, a_1), q_2 = \delta(q_1, a_2), q_3 = \delta(q_2, a_3), \dots, q_n = \delta(q_{n-1}, a_n)$$

The input string is accepted by our FSM M if the resulting sequence of states $q_0, q_1, q_2, ..., q_n$ ends in a final state; that is, is such that $q_n \in F$. This M accepts a set of strings over Σ

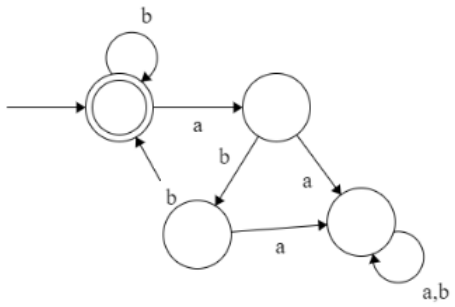
18. Give a finite state machine that accepts the set of strings over the alphabet $\{a, b, c\}$ consisting of strings with the property that there is always at most one c [5]

Solution:



19. Give a finite state machine that accepts the set of strings over the alphabet $\{a, b\}$ consisting of strings with the property that any a bust always be followed by bb [7]

Solution:



20. What is the relationship between regular expressions and finite state machines? [2]

Solution:
A set of strings is represented by a regular expression iff it is accepted by a FSM

21. Define carefully a context-free grammar and explain how one is used to generate a set of string [10]

Solution:
A grammar is a tuple (N, T, s, R) where

- T and N are disjoint finite sets of terminal and non-terminal symbols, respectively
- $s \in N$ is the start symbol

- R is a finite set of productions or rules so that

$$R \subseteq (N \cup T)^+ \times (N \cup T)^*$$

and where if $(\alpha, \beta) \in R$ when α contains at least one symbol

In a context free grammar productions are of the form

$$b \to a_1 a_2 \dots a_k, \text{ with } k > 0, b \in N, \text{ and } a_1, a_2, \dots, a_k \in N \cup T$$

A production $b \to a_1 a_2 ... a_k$ can be applied to a string ω containg b via:

$$\omega = \dots b' \underline{b} b' \cdots \Rightarrow \omega' = \dots b' a_1 a_2 \dots a_k b' \dots$$

22. What restriction on context-free grammar makes it a regular grammar, and how are regular grammars, regular expressions and finite state machines related? [5]

Solution:

Regular grammar (N,T,s,R) is a context free grammar where all productions are one of the following orms:

- $b \to a$ where $a \in T$
- $b \to ac$ where $a \in T$ and $c \in N$
- $b \to \epsilon$

A regular expression can always be written in a context free grammar, but not all context-free grammars can be expressed by a regular expression or FSM. A regular grammar can always be written as a regular expression or FSM

23. Define a regular grammar that generates the set of strings over the alphabet $\{a, b, c\}$ consisting of strings with the property that there is always at most one c [5]

Solution:

$$(N = \{s, t\}, T = \{a, b, c\}, s, R)$$

Where R is:

$$s \to \epsilon \quad s \to as \quad s \to bs \quad t \to cs \quad t \to au \quad t \to bu \quad t \to \epsilon$$

24. Define a regular grammar that generates the set of strings over the alphabet $\{a, b\}$ consisting of strings with the property that any a must always be followed by bb [7]

Solution:

$$(N = \{s, t, u, v\}, T = \{a, b\}, s, R)$$

Where R is

$$s \to \epsilon \quad u \to as \quad s \to bs \quad v \to bu \quad t \to bv \quad t \to bt \quad t \to \epsilon \quad u \to at$$

25. Which algebraic notation is normally used to specify the syntax of a programming language [1]

Solution:
Bakus Naur Form