# Revision Questions for Section 4.3 - Introduction to programming

## Nature of programming languages

4.3.1 Computer Operations
______Links: Binary Arithmetic | Boolean Logic

a. The ALU carries out the **arithmetic** operations of add and subtract. State the rules for binary addition and subtraction.
b. The ALU also takes **logical** decisions. State the rules for binary NOT, AND, OR operations.
c. Explain how the CPU also **retrieves** and **stores** data in primary memory

4.3.2 Fundamental / Compound Operations
______Links: Instruction Types | Complex Instructions

a. Give an example of a **fundamental** (basic) operation carried out by a computer's CPU.
b. Give an example of a **compound** (complex) operation (composed of many fundamental operations)

4.3.3 Language Features
______Links: Programming Language

a. Explain why a **fixed vocabulary** is an essential feature of a programming language.
b. Give an example of **ambiguity** in human language that would be difficult for a computer to process.
c. Explain in what ways **grammar rules** and **syntax** apply to a programming language.

4.3.4 Higher Level Languages
______Links: High Level Language | Low Level Language

a. Distinguish between **higher** level languages and **lower** level languages.
b. Give **three** reasons why higher level langages are needed.

4.3.5 Language Translation
______Links: Compiler | Interpreter | Comparison | JVM

a. Outine the need for a program written in a higher level language to be **translated** before it can be executed.
b. In what ways does a **compiler** differ from an **interpreter**?
c. Give three examples each of compiled and interpreted languages
d. Explain the role of the **Java Virtual Machine** in translating a Java program.

## Use of programming languages

4.3.6 Terms
______Links: Variable | Constant | Operator | Object

a. Distinguish between a **variable** and a **constant** in a programming language.
b. Define what is meant by an **'operator'** and give an example.

c. Define an **'object'** in terms of Obect-Oriented Programming.

4.3.7 Operators

Links: Relational Operators || mod | div

a. Define the **equality** operators =, ==, <>
b. Define the **inequality** operators <, <=, >, >=
c. Define the operators **mod** and **div**.

4.3.8 Variables / Constants / Operators

Links: Algorithm | Bubble Sort

a. In a recipe to make a cake, identify the **variables**, **constants** and **operators**.
b. Look at the standard **Bubble Sort** algorithm and analyse the use of variables, constants and operators.

4.3.9 Loops and Branching

Links: Loops | Conditional | FizzBuzz

a. Construct an algorithm to input a number until an **even** number has been entered.
b. Construct an algorithm to play the game *fizz-buzz* with two chosen numbers from 1 to 100.
c. Construct an algorithm to **reverse** the contents of an array.

4.3.10 Collections

Links: Collection | Array

a. Describe the main characteristics of a **collection** type of data structure.
b. Distinguish between a **collection** and an **array** data structure in Java.
c. Give three possible **applications** for a collection data structure.

4.3.11 Collection Access Methods

Links:

a. Construct an algorithm which counts the number of **occurrences** of a piece of data in a collection.
b. Construct an algorithm which adds the first 12 **multiples** of 7 to a collection called 'multiples'.
c. Construct an algorithm which removes all strings longer than **6** characters from a collection.

4.3.12 Need for subroutines and collections

Links: Subroutine | Advantages | Collection

a. Outline the need for **subroutines** (procedures) in a programmed solution.
b. Discuss the **benefits** of using subroutines in a program.
c. Outline the need for using **collections** in a large program.

4.3.13 Construct Algorithms with Subroutines, Arrays, Collections

Links:

a. Construct an algorithm to solve the *wolf-sheep-cabbage* river crossing problem using pre-defined subroutines of your choice.
b. Construct an algorithm to analyse an array of letters and tally the number of each letter **a-z** in the

array.

c. Construct an algorithm to print out the names of all students in **Year 12** or **Year 13** in a collection of *Student* objects.