Subject Specific Vocabulary

The following list provides definitions of key terms used in our GCSE Computer Science 8525 specification.

Students should be familiar with, and gain understanding from, all these terms.

Variable declaration

Variables are defined as a space in memory, given a name, assigned a value that can be changed while a program is running. Once a variable is declared it is then assigned a value, this is called initialization.

```
Name ←"Bob Smith"
```

Constant declaration

Constants are similar in definition to variables, the difference in definition being a space in memory, given a name, assigned a value that **cannot** be changed while a program is running. Constants are identified in their declaration and should be capitalized:

```
CONST PI ← 3.14
```

Assignment

Assignment is a term used to show the setting of variables to values.

```
Answer ←Num1 + Num2
```

Selection

Selection is a blanked term to refer to a programming statement that allows the changing of the flow of the program, based on a condition that is met.

```
IF Game = "won" THEN
    PRINT("you have won the game")
END IF
```

The above shows an example of nested selection, that shows if the first condition is met, then it checks the second selection statement.

Iteration

Iteration is a blanket term to refer to a programming statement that repeats code (loops). This can be both count controlled or condition controlled.

Count controlled loop

A loop that has a definite number of times to run, this can be known as definite iteration. A for loop is an example of a Count Controlled loop.

```
FOR i ← 1 TO 5
   … Instructions here …
ENDFOR
```

This can also be achieved with a WHILE loop, as long as the loop as a set number of iterations.

```
WHILE I <= 5
     …instructions here….
     I = I + 1
END WHILE
```

Condition controlled loop

Condition controlled loops are those in which the end of the loop is not known, it will continue indefinitely until the condition is met.

```
WHILE NotSolved
   … Instructions here …
ENDWHILE
```

The same effect can be achieved with a REPEAT…UNTIL loop, the benefit of which is that the program can enter the loop, gain an input then test the condition at the end, making it potentially more efficient.

```
REPEAT
   … Instructions here …
UNTIL Solved
```

Nested structures

Nesting is the process of putting a statement inside of another. This can be achieved through both iteration and selection statements. An example of nested iteration would be:

```
WHILE NotSolved
   … Instructions here …
   FOR i ← 1 TO 5
      … Instructions here …
   ENDFOR
   … Instructions here …
ENDWHILE
```

An example of nested selection would be:

```
IF GameWon THEN
   … Instructions here …
   IF Score > HighScore THEN
      … Instructions here …
   ENDIF
   … Instructions here …
ENDIF
```

Relational operator

Relational operators are those that are used to compare the relationship between 2 variables or values. This could be one of the following:

```
>, <, <=, >= , !=
```

```
eg  IF I < 6 THEN
```

Arithmetic operator

Arithmetic operators are those which are used in order to perform a mathematical function between 2 variables or values:

```
+, -, *, /
```

```
eg X = Y + Z
```

Boolean operators

Boolean operators are those which are used to test a true or false condition on variables and conditions in order to change the flow of a program:

```
eg WHILE X > 7 AND Y<6
```

both conditions would have to be true in order for the program to continue to loop.

```
WHILE X > 7 OR Y < 6
```

One or both of these conditions would have to be true in order for the program to continue to loop.

Dimensional arrays

An array is defined as multiple spaces in memory, under a single identifier in order to group elements together.

An example of defining an array is:

```
Trains ←[TrainA, TrainB, TrainC]
```

When referencing an array, this is done in the following way:

```
PRINT(Trains[0])
```

Arrays (unless referenced) start from 0.

An array of 2 Dimensions is usually referenced in a table format:

This is referenced in the following way:

```
PRINT(Trains[0,1]
```

This will identify the second record in the first row. Using the following diagram, trainB will be printed

	0	1	2
0	TrainA	TrainB	TrainC
1	TrainD	TrainE	TrainF

Records

A record is a data structure that allows multiple data types to be referenced under a single identifier. An example of a record is:

```
RECORD Car
   make : String
   model : String
   reg : String
   price : Real
   noOfDoors : Integer
ENDRECORD
```

An element of a record can be referenced in the following format:

```
Car.Make ← "Ford"
PRINT(Car.Make)
```

This will display the make of the car.

String functions

String functions are subroutines, built into the programming language in order to manipulate the string data type. There are many string functions that can be used within a language:

```
Name ← TOUPPER(Name)
```

This will turn the variable name into upper case.

Random number generation

Random Numbers can be referenced with the following notation:

```
Num ← RANDOM_INT(2,5)
```

This will display a random number between the numbers 2 and 5 and store it in the variable Num.

Subroutine (procedure/function)

A subroutine is classed as a set of instructions in order to execute a commonly used task. Subroutines are broken into 2 areas: procedures and functions. The key difference between a procedure and a function is that a procedure does not return a value whereas a function does.

Procedure	Function

`PROCEDURE Game(Value1, Value2)`
`Statements…`
`END PROCEDURE`
	`FUNCTION Game(Value1, Value2)`
	`Statements…`
	`RETURN Value3`
	`END Function`

The values in the brackets signify the parameters (variables or values) passed to the function.