

## Variables

Variables are assigned using the = operator.

```
x=3
name="Bob"
```

A variable is declared the first time a value is assigned. It assumes the data type of the value it is given.

Variables declared inside a function or procedure are local to that subroutine.

Variables in the main program can be made global with the keyword `global`.

```
global userid = 123
```

## Casting

Variables can be typecast using the `int`, `str` and `float` functions.

```
str(3) returns "3"
int ("3") returns 3
float ("3.14") returns 3.14
```

## Outputting to Screen

```
print(string)
```

Example

```
print("hello")
```

Taking Input from User

```
variable=input(prompt to user)
```

Example

```
name=input("Please enter your name")
```

## Iteration – Count Controlled

```
for i=0 to 7
    print("Hello")
next i
```

Will print hello 8 times (0–7 inclusive).

## Iteration – Condition Controlled

```
while answer!="computer"
    answer=input("What is the password?")
endwhile
```

do

```
    answer=input("What is the password?")
until answer=="computer"
```

## Logical Operators

AND OR NOT

e.g.

```
while x<=5 AND flag==false
```

## Comparison Operators

==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

## Arithmetic Operators

+	Addition e.g. $x=6+5$ gives 11
-	Subtraction e.g. $x=6-5$ gives 1
*	Multiplication e.g. $x=12*2$ gives 24
/	Division e.g. $x=12/2$ gives 6
MOD	Modulus e.g. $12\text{MOD}5$ gives 2
DIV	Quotient e.g. $17\text{DIV}5$ gives 3
^	Exponentiation e.g. $3^4$ gives 81

5

## Selection

Selection will be carried out with if/else and switch/case

if/else

```
if entry=="a" then
    print("You selected A")
elseif entry=="b" then
    print("You selected B")
else
    print("Unrecognised selection")
endif
```

switch/case

```
switch entry:
    case "A":
        print("You selected A")
    case "B":1
        print("You selected B")
    default:
        print("Unrecognised selection")

endswitch
```

## String Handling

To get the length of a string:

```
stringname.length
```

To get a substring:

```
stringname.substring(startingPosition, numberOfCharacters)
```

NB The string will start with the 0<sup>th</sup> character.

Example

```
someText="Computer Science"
```

```
print(someText.length)
```

```
print(someText.substring(3,3))
```

Will display

```
16
```

```
put
```

## Subroutines

```
function triple(number)
```

```
    return number*3
```

```
endfunction
```

Called from main program

```
y=triple(7)
```

```
procedure greeting(name)
```

```
    print("hello"+name)
```

```
endprocedure
```

Called from main program

```
greeting("Hamish")
```

**Unless stated values passed to subroutines can be assumed to be passed by value.**

**If this is relevant to the question byVal and byRef will be used. In the case below x is passed by value and y is passed by reference.**

```
procedure foobar(x:byVal, y:byRef)
```

```
    ...
```

```
    ...
```

```
endprocedure
```

## Arrays

Arrays will be 0 based and declared with the keyword *array*.

```
array names[5]
names[0]="Ahmad"
names[1]="Ben"
names[2]="Catherine"
names[3]="Dana"
names[4]="Elijah"

print(names[3])
```

Example of 2D array:

```
Array board[8,8]
board[0,0]="rook"
```

## Reading to and Writing from Files

To open a file to read from `openRead` is used and `readLine` to return a line of text from the file.

The following program makes `x` the first line of `sample.txt`

```
myFile = openRead("sample.txt")
x = myFile.readLine()
myFile.close()
```

`endOfFile()` is used to determine the end of the file. The following program will print out the contents of `sample.txt`

```
myFile = openRead("sample.txt")
while NOT myFile.endOfFile()
    print(myFile.readLine())
endwhile
myFile.close()
```

To open a file to write to `openWrite` is used and `writeLine` to add a line of text to the file. In the program below `hello world` is made the contents of `sample.txt` (any previous contents are overwritten).

```
myFile = openWrite("sample.txt")
myFile.writeLine("Hello World")
myFile.close()
```

## Comments

Comments are denoted by `//`

```
print("Hello World") //This is a comment
```

5

## Object-Oriented

---

Object oriented code will match the pseudocode listed above with the following extensions:

### Methods and Attributes:

Methods and attributes can be assumed to be public unless otherwise stated. Where the access level is relevant to the question it will always be explicit in the code denoted by the keywords.

```
public and private.  
  
private attempts = 3  
  
public procedure setAttempts(number)  
    attempts=number  
endprocedure  
  
private function getAttempts()  
    return attempts  
endfunction
```

Methods will always be instance methods, learners aren't expected to be aware of static methods. They will be called using object.method so

```
player.setAttempts(5)  
  
print(player.getAttempts())
```

### Constructors and Inheritance

Inheritance is denoted by the `inherits` keyword, superclass methods will be called with the keyword `super`. i.e. `super.methodName(parameters)` in the case of the constructor this would be `super.new()` Constructors will be procedures with the name `new`.

```
class Pet  
  
    private name  
    public procedure new(givenName)  
        name=givenName  
  
    endprocedure  
  
endclass  
  
class Dog inherits Pet  
  
    private breed  
  
    public procedure new(givenName, givenBreed)  
        super.new(givenName)  
        breed=givenBreed  
    endprocedure  
  
endclass
```

## Constructors and Inheritance

Constructors will be procedures with the name new.

```
class Pet
    private name
    public procedure new(givenName)
        name=givenName
    endprocedure
endclass
```

Inheritance is denoted by the `inherits` keyword, superclass methods will be called with the keyword `super`. i.e. `super.methodName(parameters)` in the case of the constructor this would be `super.new()`

```
class Dog inherits Pet
    private breed
    public procedure new(givenName, givenBreed)
        super.new(givenName)
        breed=givenBreed
    endprocedure
endclass
```

5

To create an instance of an object the following format is used

```
objectName = new className(parameters)
```

e.g.

```
myDog = new Dog("Fido", "Scottish Terrier")
```

## HTML

---

Learners are expected to have an awareness of the following tags. Any other tags used will be introduced in the question.

`<html>`

`<link>` to link to a CSS file

`<head>`

`<title>`

`<body>`

`<h1>` `<h2>` `<h3>`

`<img>` including the `src`, `alt`, `height` and `width` attributes.

`<a>` including the `href` attribute.

`<div>`

<form>

<input> where the input is a textbox (i.e. has the attribute `type="text"` and another attribute name to identify it) or a submit button (i.e. has the attribute `type="submit"`)

<p>

<li>

<ol>

<ul>

<script>

Any other elements used will be explained in the question.

## CSS

---

Learners are expected to be able to use CSS directly inside elements using the style attribute

```
<h1 style="color:blue;">
```

and external style sheets. In the style sheets they should be able to use CSS to define the styling of elements:

```
h1{  
  color:blue;  
}
```

classes

```
.infoBox{  
  background-color: green;  
}
```

and Identifiers

```
#menu{  
  background-color: #A2441B;  
}
```

They are expected to be familiar with the following properties.

```
background-color  
border-color  
border-style  
border-width  
color with named and hex colours  
font-family  
font-size  
height  
width
```

Any other properties used will be explained in the question.

## JavaScript

---

Learners are expected to be able to follow and write basic JavaScript code. It is hoped they will get practical experience of JavaScript in their study of the course. They will not be expected to commit exact details of syntax to memory. Questions in the exam will not penalise learners for minor inaccuracies in syntax. Learners *will* be expected to be familiar with the JavaScript equivalents of the structures listed in the pseudocode section (with the exception of input and output (see below)). They will not be expected to use JavaScript for Object Oriented programming or file handling. Questions will not be asked in JavaScript where something is passed to a subroutine by value or reference is relevant.

### Input

Input will be taken in by reading values from a form. *NB learners will not be expected to memorise the method for doing this as focus will be on what they do with that input once it is received.*

### Output

By changing the contents of an HTML element

```
chosenElement = document.getElementById("example");  
chosenElement.innerHTML = "Hello World";
```

By writing directly to the document

```
document.write("Hello World");
```

By using an alert box

```
alert("Hello World");
```

Any other JavaScript used will be explained in the question.

## Little Man Computer Instruction Set

---

In questions mnemonics will always be given according to the left hand column below. Different implementations of LMC have slight variations in mnemonics used and to take this into account the alternative mnemonics in the right hand column will be accepted in learners' answers.

| Mnemonic | Instruction        | Alternative mnemonics accepted |
|----------|--------------------|--------------------------------|
| ADD      | Add                |                                |
| SUB      | Subtract           |                                |
| STA      | Store              | STO                            |
| LDA      | Load               | LOAD                           |
| BRA      | Branch always      | BR                             |
| BRZ      | Branch if zero     | BZ                             |
| BRP      | Branch if positive | BP                             |
| INP      | Input              | IN, INPUT                      |
| OUT      | Output             |                                |
| HLT      | End program        | COB, END                       |
| DAT      | Data location      |                                |



## Structured Query Language (SQL)

Learners will be expected to be familiar with the structures below. Should any other aspects of SQL be used they will be introduced and explained in the question.

`SELECT` (including nested `SELECTs`)

`FROM` (including use of `*` wildcard)

`WHERE`

`LIKE` (with `%` used as a wildcard)

`AND`

`OR`

`DELETE`

`INSERT`

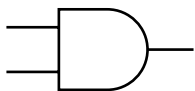
`DROP`

`JOIN` (Which is equivalent to `INNER JOIN`, there is no expectation to know about outer, left and right joins)

## Boolean Algebra

When Boolean algebra is used in questions the notation described below will be used. Other forms of notation exist and below are also a list of accepted notation we will accept from learners.

### Conjunction



Notation used:

$\wedge$  e.g.  $A \wedge B$

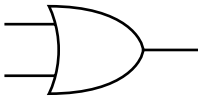
| A | B | $A \wedge B$ |
|---|---|--------------|
| T | T | T            |
| T | F | F            |
| F | T | F            |
| F | F | F            |

Alternatives accepted:

AND e.g. A AND B

e.g. A.B

## Disjunction



Notation used:

$\vee$  e.g.  $A \vee B$

| A | B | $A \vee B$ |
|---|---|------------|
| T | T | T          |
| T | F | T          |
| F | T | T          |
| F | F | F          |

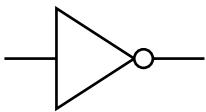
Alternatives accepted:

OR e.g. A OR B

+ e.g. A+B

5

## Negation



Notation used:

$\neg$  e.g.  $\neg A$

| A | $\neg A$ |
|---|----------|
| T | F        |
| F | T        |

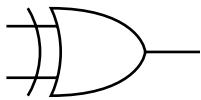
Alternatives Accepted:

*bar* e.g.  $\bar{A}$

$\sim$  e.g.  $\sim A$

NOT e.g. NOT A

## Exclusive Disjunction



Notation used:

$\underline{\vee}$  e.g.  $A \underline{\vee} B$

| A | B | $A \underline{\vee} B$ |
|---|---|------------------------|
| T | T | F                      |
| T | F | T                      |
| F | T | T                      |
| F | F | F                      |

Alternatives accepted:

XOR e.g.  $A \text{ XOR } B$

$\oplus$  e.g.  $A \oplus B$

## Equivalence / Iff

Notation used:

$\equiv$  e.g.  $(A \wedge B) \equiv \neg(\neg A \vee \neg B)$

Alternatives accepted:

$\leftrightarrow$