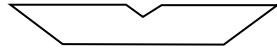# APPENDIX A

Conventions followed in specification

Note 1

Where Von Neumann architecture is represented diagrammatically, the following symbols are used:

Arithmetic logic unit

Register

Control unit

Note 2

Where candidates are required to apply computing-related mathematics, the following arithmetical and relational operators will be used:

Operator	Meaning	Example
>	Greater than	A>B will return TRUE if the value of A is higher than the value of B otherwise it will return FALSE.
<	Less than	A<B will return TRUE if the value of A is lower that the value of B otherwise it will return FALSE.
<=	Less than or equal to	A<=B will return TRUE if A is the same as or lower than B otherwise it will return FALSE.
>=	Greater than or equal to	A>=B will return TRUE if A is the same as or higher than B otherwise it will return FALSE.
<>	Not equal to	A<>B will return TRUE if A is not the same as B but FALSE if A is the same as B.
EQUALS (usually =)	The same as	A=B will return TRUE if A is the same as B otherwise it will return FALSE.
AND	Both statements must be true for the argument as a whole to be true.	(A=1) AND (B=4) will return TRUE if A is 1 and B is 4. It would return FALSE in all other situations.
OR	Only one of the statements needs to be true for the argument as a whole to be true.	(A=1) OR (B=4) will return TRUE if A is 1 or B is 4. It would only return FALSE if A is not 1 and B is not 4.
NOT	The opposite of	NOT(A) will return TRUE if A is FALSE and FALSE if A is TRUE.

XOR	The argument is false if both statements are true. The argument is false if both statements are false. Otherwise the statement is true.	A XOR B would return TRUE if A and B are different values.
DIV	Integer division Finds the quotient or the 'whole number of times' a divisor can be divided into a number.	11 DIV 2 = 5 The quotient is 5 as 2 divides into 11 a whole number of 5 times
MOD	Modulo division Finds the remainder when a divisor is divided into a number.	11 MOD 2 = 1 The remainder is 1 as 2 divides 5 times into 11 with '1 remaining'

Note 3

Algorithms written in pseudo code will be represented using the following convention:

Construct	Example usage
Declare subroutines	`Declare CapitalLetterOfName` `End Subroutine`
Call a subroutine	`call SubroutineNeeded`
Declare and use arrays	`myarray[99]`
Literal outputs	`output "Please enter a number"`
Variable names	`myvariable`
Define variable data type	`myvariable is integer`
Data types	`integer, character, string,` `boolean, real`
Assignment	`set counter = 0`
Selection	`if . . . else . . . end if`

Indent at least single space after if or do or repeat etc.	`if counter = 1`
	` output counter`
	`end if`

Annotation	`{Some annotation goes here}`
Comments	`/** Comments for program */`

Repetition	`for i . . . next i`
	`repeat . . . until`
	`do . . . loop`
	`do . . . while`
	`while . . . repeat`

| String handling | ```
mid(string,x,y)
left(string,x)
right(string,x)
instring(x,stringa,stringb)
len(string)
val(string)
int(string)
trim(string)
char(number)
``` |
|---|---|

Logical operators `AND, OR, NOT and XOR` will be in upper case.
Logical `TRUE` and `FALSE` will be in upper case.

Note 4

Algorithms represented using a flowchart will use the following convention:

Start / Stop procedure

Decision box

Input / Output

Operation

Connector

Store / Subroutine call

Flow of control
(Arrowhead indicates
direction of flow)