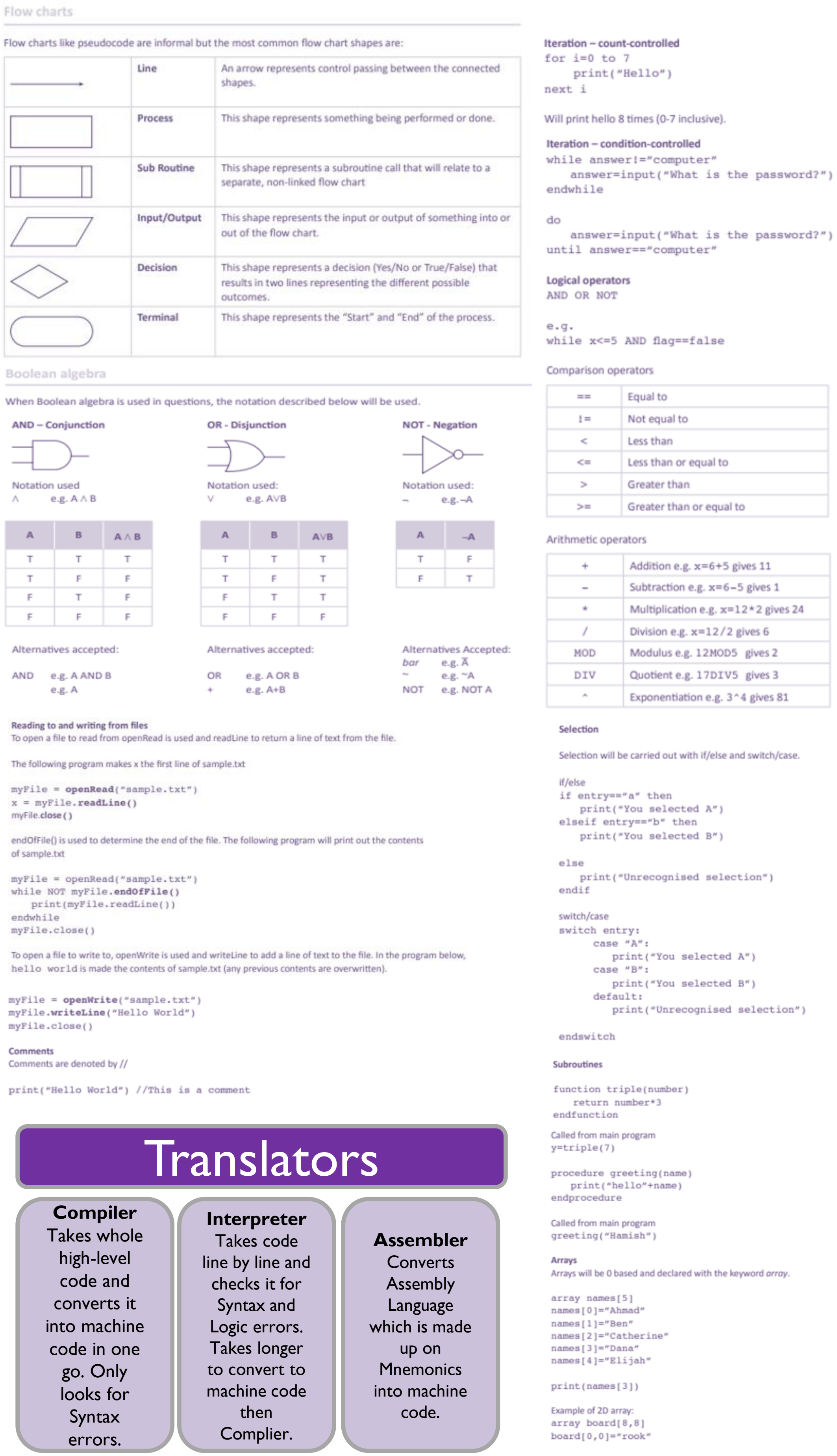
COMPUTER SCIENCE LEARNING MAT



Flow charts

Flow charts like pseudocode are informal but the most common flow chart shapes are:

	Line	An arrow represents control passing between the connected shapes.
	Process	This shape represents something being performed or done.
	Sub Routine	This shape represents a subroutine call that will relate to a separate, non-linked flow chart
	Input/Output	This shape represents the input or output of something into or out of the flow chart.
	Decision	This shape represents a decision (Yes/No or True/False) that results in two lines representing the different possible outcomes.
	Terminal	This shape represents the "Start" and "End" of the process.

Boolean algebra

When Boolean algebra is used in questions, the notation described below will be used.

AND – Conjunction

Notation used
∧ e.g. A ∧ B

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

Alternatives accepted:

AND e.g. A AND B
e.g. A

OR - Disjunction

Notation used:
∨ e.g. A∨B

A	B	A∨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

NOT - Negation

Notation used:
¬ e.g. ¬A

A	¬A
T	F
F	T

Alternatives Accepted:
bar e.g. $\overline{A}$
~ e.g. ~A
NOT e.g. NOT A

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
```

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. 12MOD5 gives 2
DIV	Quotient e.g. 17DIV5 gives 3
^	Exponentiation e.g. 3^4 gives 81

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":
        print("You selected B")
     default:
        print("Unrecognised selection")

endswitch
```

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")
```

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"
```

Translators

Compiler
Takes whole high-level code and converts it into machine code in one go. Only looks for Syntax errors.

Interpreter
Takes code line by line and checks it for Syntax and Logic errors. Takes longer to convert to machine code then Complier.

Assembler
Converts Assembly Language which is made up on Mnemonics into machine code.

SYSTEMS ARCHITECTURE

MAR = Memory Address Register
MDR = Memory Data Register
CIR = Current Instruction Register
PC = Program Counter

MEMORY

RAM's memory is volatile it is temporary

STORAGE

For magnetic, optical and solid state consider

- Capacity
- Cost
- Durability
- Portability
- Speed
- Reliability

WIRED AND WIRLESS NETWORKS

The Internet is not the World Wide Web, the Internet is simply a network of networks.

ALGORITHMS

Computational Thinking at GCSE level is understanding:-

- Abstraction
- Decomposition
- Pattern Recognition
- Algorithms

DATA REPRESENTATION

You need to be able to:-

- Convert between denary, binary and hexadecimal.
- Add two binary numbers together.
- Explain compression including lossy and lossless.
- Understand how to represent characters, sound and images.

COMP 1 Computer Systems

Systems Architecture

Memory

Storage

Wired & Wireless Networks

Network topologies and layers

System security

System software

COMP 2 Computational thinking, algorithms & programming

Algorithms

Programming techniques

Producing robust programs

Computational logic

Translators and facilities of languages

Data representation

COMPUTER SCIENCE COMMAND WORDS

Add: Join something to something else so as to increase the size, number, or amount.
Analyse: Break down in order to bring out the essential elements or structure. To identify parts and relationships, and to interpret information to reach conclusions.
Annotate: Add brief notes to a diagram or graph.
Calculate: Obtain a numerical answer showing the relevant stages in the working.
Compare: Give an account of the similarities and differences between two (or more) items or situations, referring to both (all) of them throughout.
Complete: Provide all the necessary or appropriate parts.
Convert: Change the form, character, or function of something.
Define: Give the precise meaning of a word, phrase, concept or physical quantity.
Describe: Give a detailed account or picture of a situation, event, pattern or process
Design: Produce a plan, simulation or model.
Discuss: Offer a considered and balanced review that includes a range of arguments, factors or hypotheses. Opinions or conclusions should be presented clearly and supported by appropriate evidence.
Draw: Produce (a picture or diagram) by making lines and marks on paper with a pencil, pen, etc.
Evaluate: Assess the implications and limitations; to make judgments about the ideas, works, solutions or methods in relation to selected criteria.
Explain: Give a detailed account including reasons or causes.
Give: Present information which determines the importance of an event or issue. Quite often used to show causation.
How: In what way or manner; by what means.
Identify: Provide an answer from a number of possibilities. Recognise and state briefly a distinguishing factor or feature.
Justify: Give valid reasons or evidence to support an answer or conclusion.
Label: Add title, labels or brief explanation(s) to a diagram or graph.
List: Give a sequence of brief answers with no explanation.
Order: Put the responses into a logical sequence.
Outline: Give a brief account or summary.
Show: Give steps in a derivation or calculation.
Solve: Obtain the answer(s) using algebraic and/or numerical and/or graphical methods.
State: Give a specific name, value or other brief answer without explanation or calculation.
Tick: Mark (an item) with a tick or select (a box) on a form, questionnaire etc. to indicate that something has been chosen.
What: Asking for information specifying something.
Write/Rewrite: Mark (letters, words, or other symbols) on a surface, typically paper, with a pen, pencil, or similar implement/Write (something) again so as to alter or improve it.