Section 09.2 - Algorithms

Layer 6: High-Order Language

Syllabus Content Section 09: Algorithm Design and Problem Solving

S09.2.1 Show understanding that an algorithm is a solution to a problem expressed as a sequence of defined steps

---

`Algorithm`: a sequence of defined steps that can be carried out to perform a task

1. How you do something
2. The steps you need to do
3. Performing a task
4. A sequence of steps
5. A sequence of steps that can allow you to perform a task
6. Performing a task in order

```ad-note
title: S09.2.2 Use suitable identifier names for the representation of data used by a problem and represent these using an identifier table
collapse: open

---

> Examples

- variable represent name
    - Stu1Name
    - Stu2Name
- variable represent age
    - BobAge
    - TomAge
```

`Identifier table`: a table listing the variable identifiers required for the solution, with explanations and data types

Identifier	Explanation
BiggestSoFar	Stores the biggest number input so far
NextNumber	The next number to be input
Counter	Stores how many numbers have been input so far

~~~
```ad-note
title: S09.2.3 Write pseudocode that contains input, process and output
collapse: open

---

```js
// Declare variable
DECLARE Score1:INTEGER
DECLARE Score2:INTEGER
DECLARE Total:INTEGER
// input
INPUT Score1
INPUT Score2
//process
Total <- Score1 + Score2
// output
OUTPUT "Your total score is: ", Total
~~~

S09.2.4 Write pseudocode using the three basic constructs of sequence, selection and iteration (repetition)

`Assignment`: a value is given a name (identifier) or the value associated with a given identifier is changed.
`Sequence`: a number of steps are performed, one aft er the other.
`Selection`: under certain conditions some steps are performed, otherwise diff

erent (or no) steps are performed.
`Repetition`: a sequence of steps is performed a number of times. This is also known as iteration or looping.

> sequence

run the code line by line

> Selction

- IF statements
- Case Statements

> Iteration (repetition)

- Count-controlled (FOR) loops
- Post-condition (REPEAT) loops
- Pre-condition (WHILE) loops

S09.2.5 Document a simple algorithm using pseudocode

> two number addition

```
FUNCTION Add(Num1:INTEGER, Num2:INTEGER) RETURNS INTEGER
        DECLARE Total: INTEGER
        Total <- Num1 + Num2
        RETURN Total
ENDFUNCTION
```

S09.2.6 Write pseudocode from:

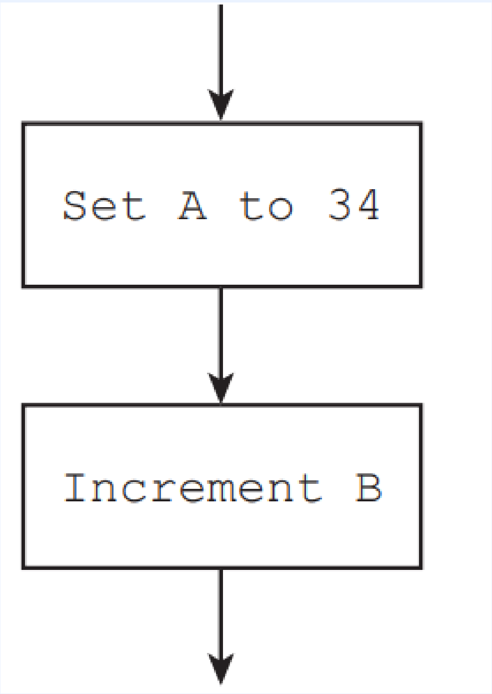
- a structured English description
- a flowchart

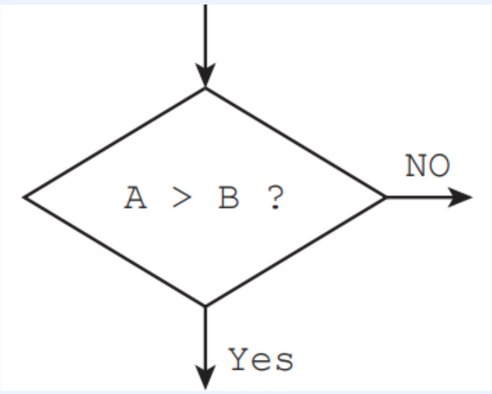
`Structured English`: a subset of the English language that consists of command statements used to describe an algorithm

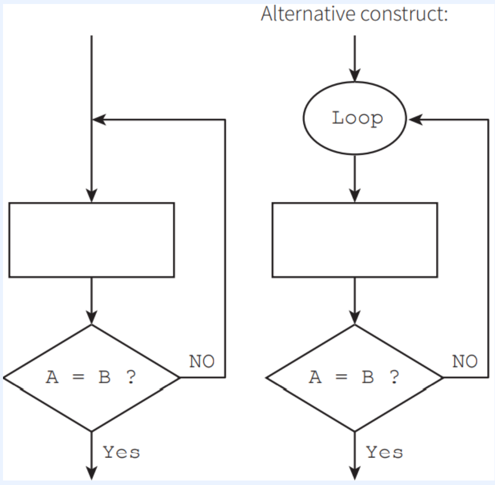
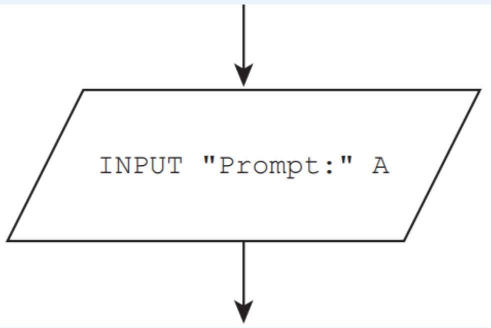
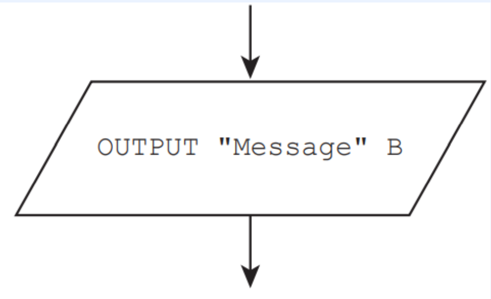
`Pseudocode`: a way of using keywords and identifiers to describe an algorithm without following the syntax of a particular programming language

`Flowchart`: shapes linked together to represent the sequential steps of an algorithm

	Structured English	Pseudocode	Flowchart

Assignment and Sequence	SET A TO 34
	INCREMENT B
		A <- 34
		B <- B + 1
			

Selection	IF A IS GREATER THAN B THEN ... ELSE ...	IF A > B THEN ...
		ELSE ...
		ENDIF
			

	Structured English	Pseudocode	Flowchart
Repetition	REPEAT UNTIL A IS EQUAL TO B …	REPEAT … UNTIL A = B	 Alternative construct: Loop A = B ? NO Yes A = B ? NO Yes
Input	INPUT A	INPUT "Prompt: " A	
Output	OUTPUT "Message" OUTPUT B	OUTPUT "Message", B	

S09.2.7 Describe and use the process of stepwise refinement to express an algorithm to a level of detail from which the task may be programmed

`Stepwise refinement`: breaking down the steps of an outline solution into smaller and smaller steps

> Example

Use stepwise refinement to output a hollow triangle. For example the two input values A and 9 result in the following output:

```
    A
   A A
  A   A
 A     A
AAAAAAAAA
```

> A first attempt at solving this problem using structured English is:

```
Set up initial values
REPEAT
    Output leading number of spaces
    Output symbol, middle spaces, symbol
    Adjust number of spaces and number of symbols to be output in next row
UNTIL the required number of symbols have been output in one row
```

S09.2.8 Use logic statements to define parts of an algorithm solution

---

Operator	Comparison
=	Is equal to
<	Is less than
>	Is greater than
<=	Is less than or equal to
>=	Is Greater than or equal to
<>	Is not equal to

> Examples

- If Age < 13 then person is a child.
- If Age > 19 then person is an adult.
- If Age >= 13 AND Age <= 19 then person is a teenager