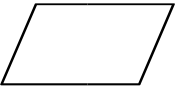
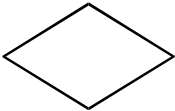
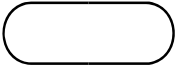
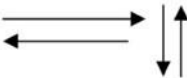


Notes on Flow Charts and Pseudocode

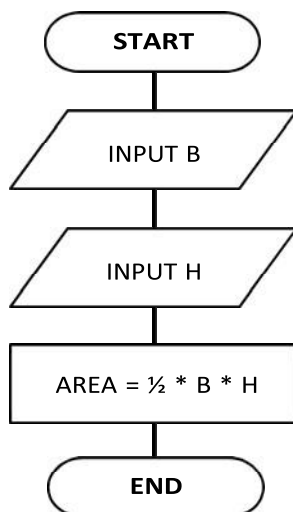
Flow Charts

These are sequences of instructions linked together using different-shaped boxes to explain the control flow. The following symbols are used in flow chart diagrams to show the steps through an algorithm or program:

	Describes a process or operation in the program
	Indicates the input(s) or output(s) of data in the program
	In a decision diamond, there will be an input and two outputs where one represents true and the other false
	Connecting circles are used to link parts of a flow chart that are drawn on multiple pages
	This symbol is used to indicate that a pre-defined subroutine has been used
	This lozenge-shaped symbol indicates the start or end of a program
	Flow lines and arrows indicate the direction of flow through the program

Example:

The flow chart below shows the process of calculating the **area of a right-angled triangle** where the height and base of the triangle are input and the area is then calculated.



The pseudocode solution for the same problem is shown below.

Note that where necessary it is easier to modify or adapt the text-based solution.

Start

Input Base

Input Height

Area $\leftarrow \frac{1}{2} * \text{Height} * \text{Base}$

Output "Right-angled Triangle Area"; Area

End

Drawing Flow Charts

Flow charts can be drawn freehand using pen and paper, or by using office software such as MS PowerPoint or MS Word. There are also several specialist flow chart design tools that are available, including:

- <http://www.gliffy.com/>
- <http://www.nchsoftware.com/chart>
- <http://www.smartdraw.com/software/flowchart-software.htm>

Limitations of Flow Charts

One of the main advantages of using a flow chart is that it offers a pictorial method of representing an algorithm and includes the flow of control.

However, there are limitations in using flow charts, as described below:

- The reproduction of a flow chart can be a problem, as the flow chart symbols and included text cannot be entered by simply typing or word processing.
- Flow chart updates are often required due to modifications for the solution; in some cases the flow chart will need redrawing completely.
- Complex logic can be complicated and difficult to understand if it spans more than a single-page flow chart.
- Flow charts may have no relationship to the actual program code and so do not form a useful method of documentation; for example, a CASE or SWITCH statement is a more effective construct than the use of many repeated IF–THEN–ELSE statements.

Pseudocode

This is a written list of steps that are required to complete a process, not connected to any particular programming language. Note that pseudocode can be written in many styles, but it should be in sufficient detail to allow the developer to write code based on it.

Data Types

Data Type	Explanation	Example
INTEGER	Whole number	2014
STRING	Sequence of characters such as: letters, spaces, numbers	"the eagle has landed"
FLOAT	A number with a fractional part	24.65
BOOLEAN	Can have only logical data	True or False

Arrays

Typical array declarations:

```
INTEGER Years[4]
STRING Names[5]
```

All elements of the Years array are shown initialised below:

```
Years ← [2015, 2014, 2013, 2012]
```

Operators

Operator	Explanation	Example
+, -, *, /	Arithmetic operators – brackets can be used to make precedence obvious	14 + 5 (15 + x) / 2
AND, OR, NOT	Boolean operators – brackets can be used to make precedence obvious	X ← (A AND B) OR C
=, >, >=, <, <=, <>	Logic comparison operators	WHILE (X >= 2)
!=, ==	Comparison operators for strings	IF (W == "Finished") THEN

Assignments

These are made using the '←' symbol; for example: **X ← 15**

Comments

These are made using the '#' symbol.

Sequencing

In a sequence structure the program performs each action or statement in order; it is not possible to branch or skip an action.

```
Input Length
Input Width
Area ← Length * Width
Output "Rectangle"; Area
```

Selection

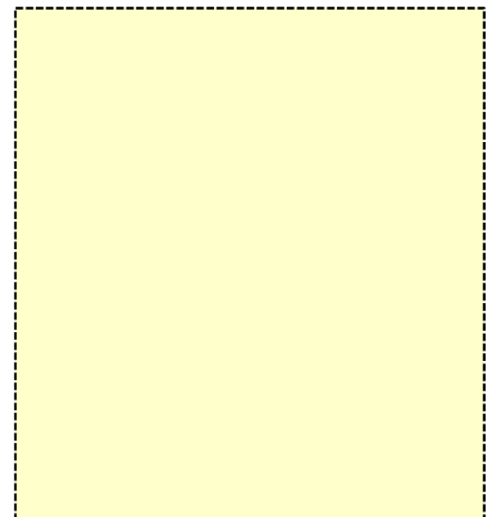
A selection structure is where the program executes different actions or statements depending on the value of a variable.

IF–THEN statements are used to execute one block of code when a Boolean condition is TRUE; there is no alternative branching when the Boolean condition is FALSE.

```
IF (X < Y) THEN
    High ← 100
ENDIF
```

IF–THEN–ELSE statements are used to execute one block of code when a Boolean condition is TRUE, and an alternative when the Boolean condition is FALSE.

```
IF (X > Y) THEN
    High ← X
ELSE
    High ← Y
ENDIF
```



CASE statements (termed SWITCH statements in some programming languages) are a variation of an IF–THEN–ELSE statement where several IFs are required in a row.

```
CASE Exam Grades
    BETWEEN 0 AND 30:
THEN grade ← F
    BETWEEN 31 AND 40:
THEN grade ← F
    BETWEEN 41 AND 50:
THEN grade ← F
    BETWEEN 51 AND 60:
THEN grade ← F
    BETWEEN 61 AND 70:
THEN grade ← F
    BETWEEN 71 AND 100:
THEN grade ← F
END CASE
```

Iteration

Iteration (also known as repetition) is where a program executes a statement or statements that are repeated until some logical condition is satisfied.

FOR... NEXT LOOP

This iterative control method is useful when the same instructions or calculations have to be carried out for a known number of iterations.

```
FOR X = 1 TO 10
    OUTPUT Variable[X]
ENDFOR
```

DO... WHILE LOOP

In this iterative control method technique, the loop operates when a condition is satisfied at the start of the loop and stops when this is no longer true.

```
F ← 1
counter ← 10
WHILE counter > 0
    F ← F + 1
    counter ← counter - 1
ENDWHILE
```

REPEAT... UNTIL LOOP

In this iterative control method technique, the loop operates at least once; then the condition is tested at the end of the loop and repeats until the condition is no longer true.

```
REPEAT
    OUTPUT "Enter a Number"
    N ← INPUT
    OUTPUT N
UNTIL N = 10
```

File Handling

File handling commands are shown below:

- **OPEN (<filename>, flag)** – opens a filename where the flag can be set to reading (r), writing (w) or appending (a)
- **READLINE (<filename>, n)** – reads line number *n* from an existing file
- **WRITELINE (<filename>, n, value)** – writes/overwrites a value into line *n* of an existing file
- **CLOSE (<filename>)** – closes the specified file

Defining Functions

The following example shows a method of defining a function in pseudocode.

```
FUNCTION CLEAR (Names)
    # Zero all the elements of an array
    FOR K = 1 TO 100
        Names[K] ← 0
    ENDFOR
ENDFUNCTION
```

This function can be called by: **CLEAR(X)** where X is the array that needs to be zeroed.

Functions

Functions are named in upper-case letters, with the variable that will be returned in brackets. Some typical functions available in most languages include:

- **LEN(X)** – X is returned with the length of an array or a string
- **MAX(X)** – X is returned with the largest of a set of numbers
- **RANDOM(x, y)** – a random number in a given range (x, y)
- **SLEEP(X)** – sleep or pause function where time paused is given in seconds
- **ISDIGIT(X)** – tests whether character X is a number
- **ISUPPER(X)** – tests whether character X is an upper-case letter
- **ISLOWER(X)** – tests whether character X is a lower-case letter

Input/Output

The following examples show how data is input and output using pseudocode.

- **OUTPUT** sends a message to the screen or printer.

OUTPUT "Do you wish to continue, Y / N"

- **INPUT** receives an input from a device, such as a temperature sensor.

Room Temperature ← INPUT

- **USERINPUT** receives an input from a user, normally via a keyboard.

First Name ← USERINPUT

Use of Pseudocode

There are some disadvantages in using pseudocode, such as: there is no standardised style; a beginner to programming finds it technically more difficult to write in comparison with drawing a flow chart. However, there are benefits to using pseudocode, as detailed below:

- Plain text – so can be easily written and modified where necessary in any word processor
 - Easy to read and understand by non-programmers as well as experts
 - x Structured design, but not an actual programming language, so developer can use it to easily create a solution in a suitable programming language
-

Worked Example – George's Greenhouse

Scenario

George's Greenhouse

George is the caretaker at school, and he has been given a greenhouse to grow plants for the school grounds. He would like to keep the greenhouse at a set temperature so that plants grow quickly; he would like to be able to change the set temperature between 17 °C and 26 °C, depending on what he is growing.

He has bought a greenhouse control kit online, which consists of a heater and a wireless temperature sensor, which can be connected to his computer.

Temperature Sensor Note:

In this case a temperature sensor is a device that will gather data concerning the temperature of the greenhouse, and convert it into a form that can be understood by the computer control system.

Your
Goal

Design an algorithm to help George set up an automatic greenhouse control system.



© ZigZag Education, 2016

Tasks

Complete the following tasks in order to develop an algorithm to solve the problem.

A . ANALYSIS

Analyse the scenario and identify the inputs, processes and outputs needed for the algorithm.

B . FLOW CHART

Represent the algorithm using a flow chart.

C . PSEUDOCODE

Represent the algorithm using pseudocode.



Extension

**Using a suitable programming language, code your solution.
Test the program using suitable data.**

Task A: Analysis

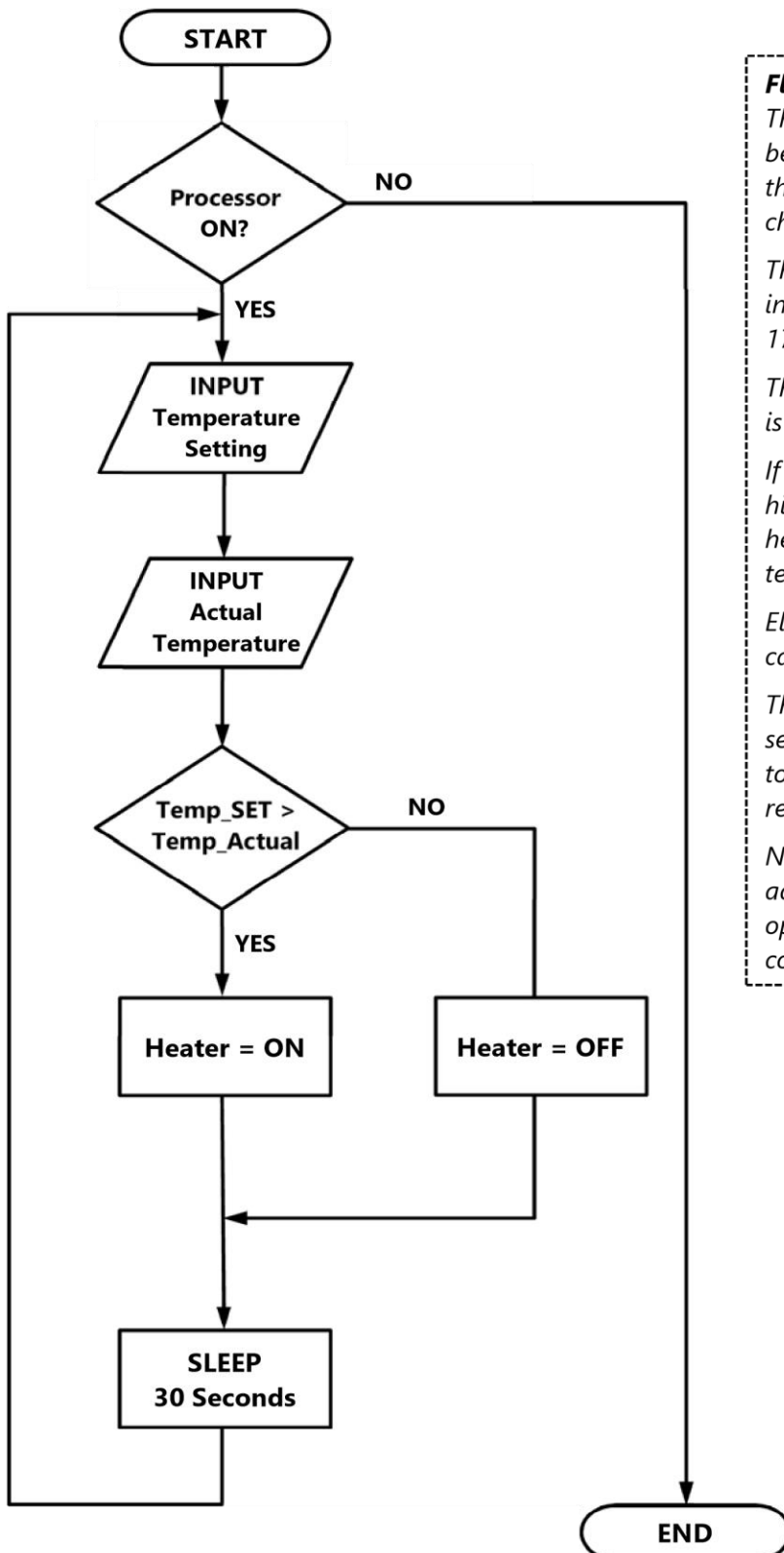
Analyse the scenario and identify the inputs, processes and outputs needed for the algorithm.

Hint: consider the mathematical calculation(s) that the algorithm is required to perform.

Inputs	Outputs	Process
Control Computer Powered Logic (Processor = ON or OFF) Actual Temperature (Temp_Actual) Temperature Control Setting (Temp_Set)	Heater (ON/OFF)	While the computer control system is switched on, the actual temperature of the greenhouse will be checked against the temperature setting. The heater will be switched ON if the greenhouse is too cold or The heater will be switched OFF if the greenhouse is too hot This process will be continued, although a time delay will be added to give the system time to respond.

Task B: Flow Chart

Represent the algorithm using a flow chart.



Flow Chart Commentary

The school greenhouse temperature has been computer controlled. Therefore if the processor is not powered on, the flow chart will go straight to the end.

The control temperature setting is input into the system; it would be in the range 17 to 27 °C.

The actual temperature from the sensor is then input into the system.

If the control temperature setting is higher than the actual temperature, the heater is switched ON to raise the temperature of the greenhouse.

Else the heater is switched OFF, in which case the greenhouse will cool down.

There is then a pause (or sleep) for 30 seconds to allow the system to respond to the heater input, before the process is repeated.

Note this is a basic system; more advanced systems might include open/close window motors or computer-controlled fans.

Marking Guidance (7 Marks)

- ✓ 2 marks for input statements
- ✓ 2 marks for flow chart structure
- ✓ 3 marks for heater control logic

Alternative working solutions should be given credit.

Task C: Pseudocode

Represent the algorithm using a pseudocode.

START

```
# Repeat process while computer control processor is operating
WHILE (Processor = ON) THEN
    # Read control setting (17 to 27 °C) and actual temperature
    # Inputs from temperature sensors

    Temp_Set ← INPUT
    Temp_Actual ← INPUT
    # Greenhouse needs to heat up
    IF (Temp_Set > Temp_Actual) THEN
        Heater ← ON
    # Greenhouse needs to cool down
    ELSE
        Heater ← OFF
    ENDIF
    # Time delay for system to respond
    SLEEP (30)
ENDWHILE
```

END

Marking Guidance (10 Marks)

- ✓ 2 marks for input statements
- ✓ 2 marks for use of WHILE loop
- ✓ 3 marks for heater control logic
- ✓ 3 marks for appropriate comment statements

Alternative working solutions should be given credit.