# OCR GCSE Computer Science

Your notes

## Designing, Creating & Refining Algorithms

### Contents


Your notes

# Inputs, Processes, & Outputs for a Problem

- Applying algorithmic thinking leads to a set of precise step-by-step instructions that can solve a problem
- To create an algorithm, the **inputs**, **processes** and **outputs** must be identified

## What is an input?

- An input is **data or information being entered/taken into a program** before it is processed in the algorithm
- An input can come from a variety of sources, such as:
  - **User** - keyboard, mouse, controller, microphone
  - **Sensors** - temperature, pressure, movement

## What is a process?

- A process is **a doing action performed in the algorithm** that transforms inputs into the desired output. The central processing unit (**CPU**) executes the instructions that define the process
- An example would be:
  - **Comparing two numbers**
  - **Calculating an average**

## What is an output?

- An output is **the result of the processing in an algorithm** and usually the way a user can see if an algorithm works as intended
- An output can take various forms, such as:
  - **Numbers** - result of calculations
  - **Text**
  - **Images**
  - **Actions** - triggering events

## Example 1 - Area of a shape

- A user wants to write a program to calculate the area of a shape

| Input | Process | Output |
|---|---|---|
| ▪ Length<br>▪ Width | ▪ Length X width | ▪ Area |

Your notes

## Example 2 - Average test score

- A teacher wants to calculate the average mark achieved on a test amongst students in a class. The teacher needs to enter how many students in the class and for each students a score out of 50

| Input | Process | Output |
|---|---|---|
| ▪ Number of students<br>▪ Score per student | ▪ TotalScore = TotalScore + score per student<br>▪ Average = TotalScore / Number of students | ▪ Average mark |

### Worked Example

A bus company offers a discount to passengers if they have a valid 'student' card or are over 65 years of age.

Identify all the **inputs** that will be required in an algorithm to solve this problem **[2]**

**Answer**

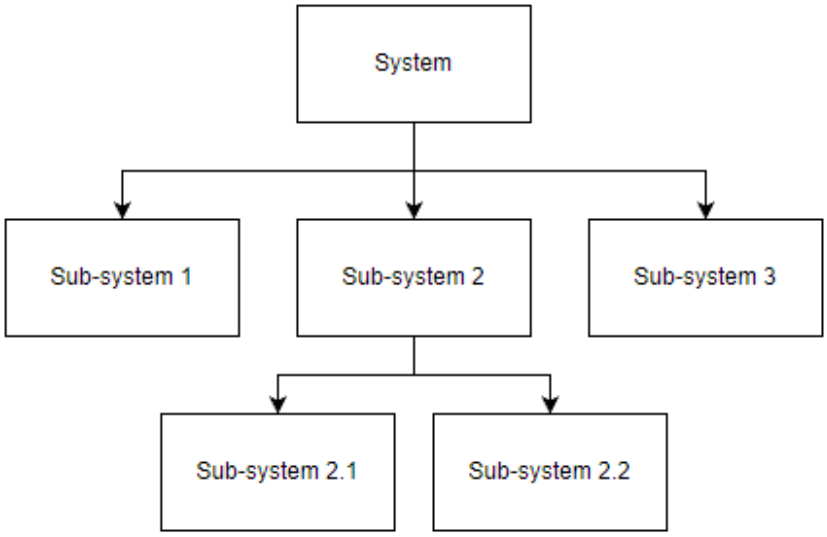
- Student card (YES/NO)
- Age (integer)

Your notes

# Structure Diagrams

## What is a structure diagram?

- A structure diagram is a **visual representation of problem decomposition**
- A **tool** to show how a complex problem can be **broken down** into more manageable sub problems
- A **planning tool** for developers during the **analysis** of a problem




### Worked Example

A hairdressers uses a mobile phone app to allow clients to book appointments.

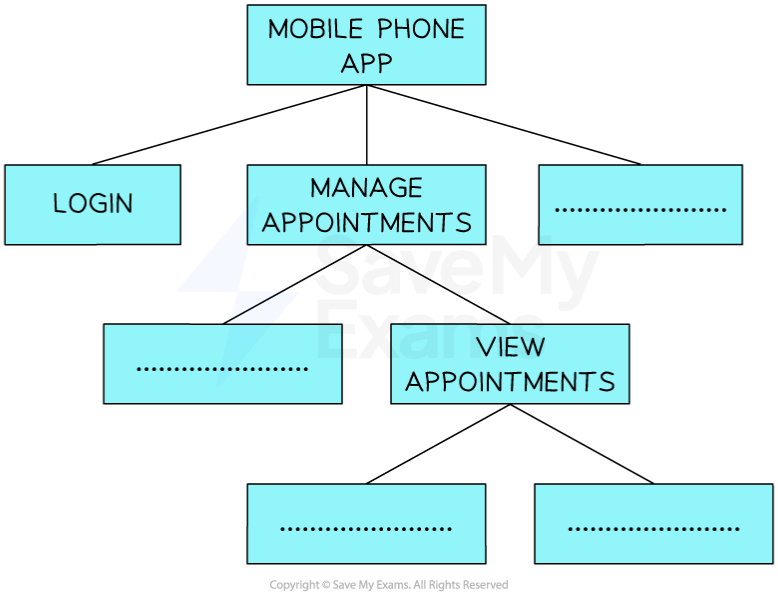
Clients must log in before they can use the system. They then choose to book a new appointment, view all appointments already made or update their personal details. If clients choose to view their appointments, they can either view them on-screen or print them off.

Your notes

A structure diagram has been used to design the mobile phone app.

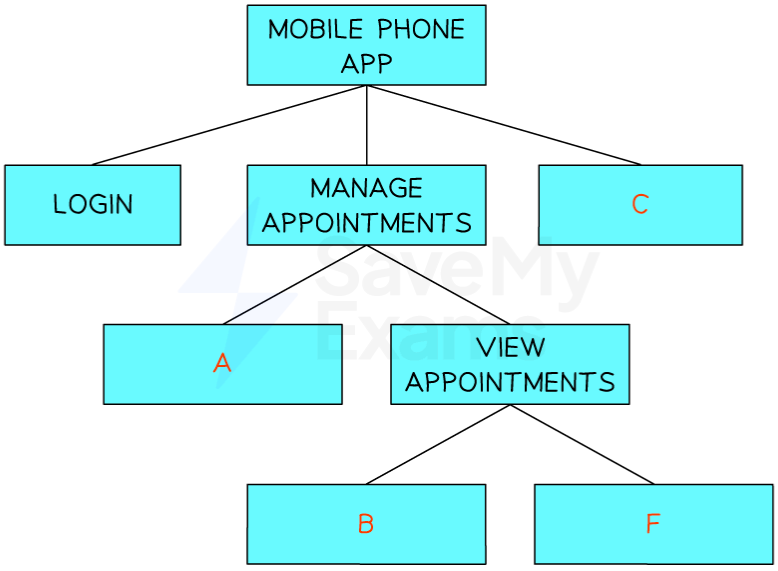
Write one letter from the following table in each space to complete the structure diagram **[4]**

| Letter | Task |
|---|---|
| **A** | Book new appointment |
| **B** | View appointments on screen |
| **C** | Update personal details |
| **D** | Check price list |
| **E** | Log out of system |
| **F** | Print a paper copy of appointments |





**Answer**






Your notes

## Pseudocode & Flowcharts


Your notes

- When designing algorithms there are two main tools that can be used to describe them:
  - **Pseudocode**
  - **Flowcharts**

### Examiner Tips and Tricks

Remember, in the exam you will be expected to **create**, **interpret**, **correct** and **refine** algorithms in either flowcharts, pseudocode or OCR exam reference language

# Pseudocode

## What is pseudocode?

- Pseudocode is a **text based tool** that uses short **English words/statements** to **describe an algorithm**
- Pseudocode is **more structured** than writing sentences in English, but is **very flexible**

## Example

- A casino would like a program that asks users to **enter an age**, if they are 18 or over they can enter the site, if not then they are given a suitable message

| Pseudocode |
| --- |
| INPUT age<br>IF age >= 18 THEN<br>OUTPUT "Welcome to the site"<br>ELSE<br>OUTPUT "Sorry, this site is for users 18 and over"<br>END IF |

- The casino would like the algorithm **refined** so that the user also **enter their first name** and this is used to greet the user when they access the site

| Pseudocode |
| --- |
| INPUT fname<br><br>INPUT age<br><br>IF age >= 18 THEN<br><br>OUTPUT ("Welcome to the site", fname)<br><br>ELSE<br><br>OUTPUT "Sorry, this site is for users 18 and over"<br><br>END IF |


Your notes

## What is OCR exam reference language?

- OCR exam reference language is the **officially designed pseudocode** that is seen in **OCR based exams** to describe algorithms
- Pseudocode has no official syntax so to keep exams consistent OCR have developed their own

### Examples

| Function | OCR exam reference language |
| --- | --- |
| **OUTPUT** | print("Hello") |
| **INPUT** | num = input("Enter a number") |
| **SELECTION** | if num == 2 then<br><br>...<br><br>elseif num < 4 then<br><br>...<br><br>endif |

| | |
|---|---|
| **FOR LOOPS** | for i = 1 to 10<br>...<br>next i |
| **WHILE LOOPS** | while (i != 11)<br>...<br>endwhile |

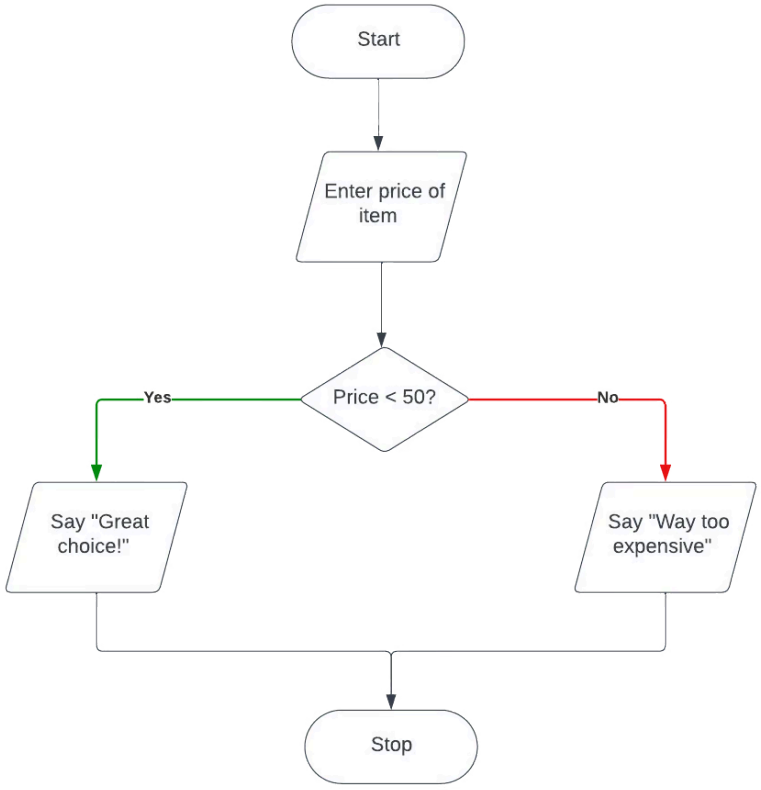

Your notes

## Examiner Tips and Tricks

OCR exam reference language is so close to Python syntax that you can write algorithms in Python in both section A and B of the exam!

## Worked Example

Your notes




Rewrite the flowchart as a pseudocode **[4]**
You must use **either**

- Pseudocode **or**
- OCR Exam Reference Language

**Answer**

| Pseudocode | OCR exam reference language |
|---|---|

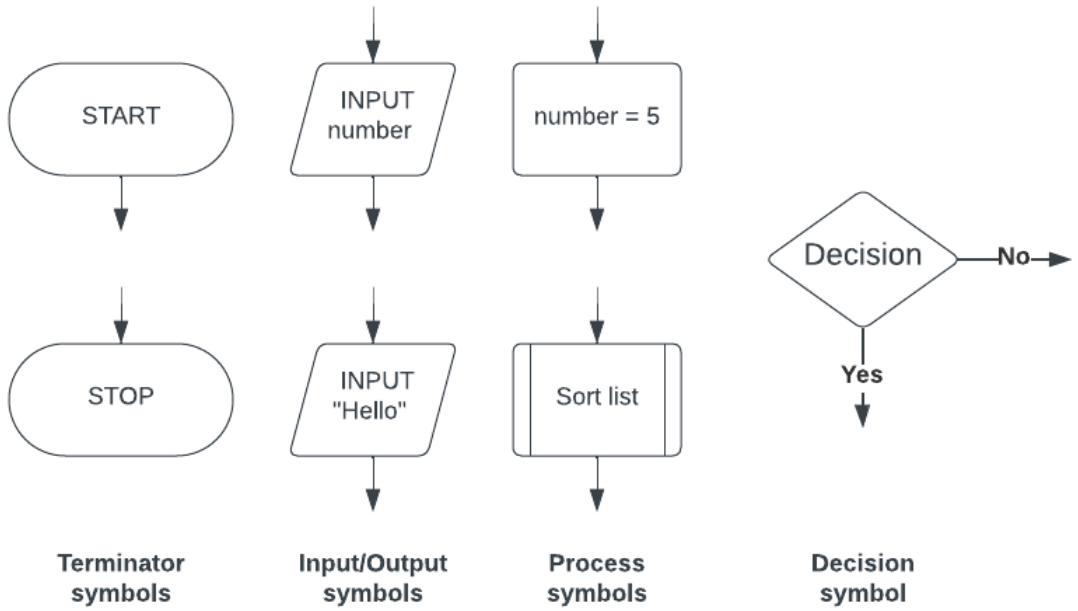
| | |
|---|---|
| INPUT price | price = input("Enter price") |
| IF price < 50 THEN | if price < 50 then |
| OUTPUT "Great choice!" | print("Great choice!") |
| ELSE | else |
| OUTPUT "Way too expensive" | print("Way too expensive") |
| END IF | end if |

Your notes

# Flowcharts

## What is a flowchart?

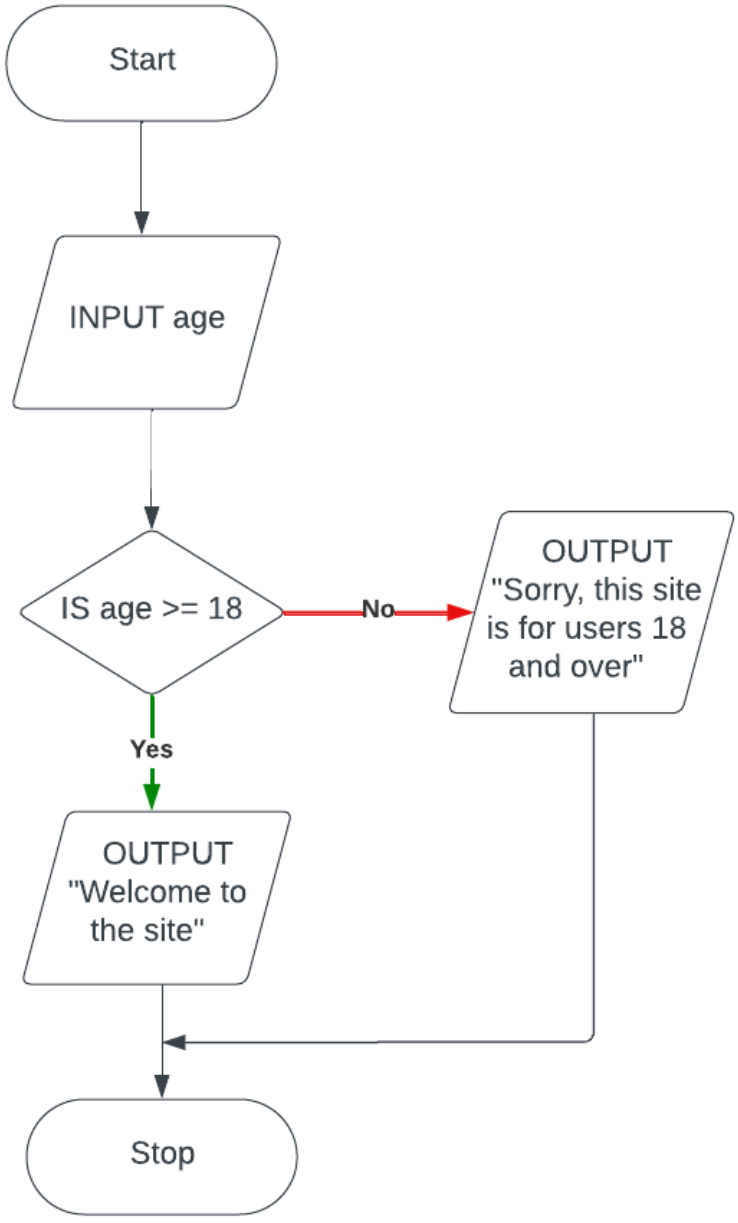
- Flowcharts are a **visual tool** that uses **shapes to represent different functions** to **describe an algorithm**
- Flowcharts show the data that is **input** and **output**, the **processes** that take place and any **decisions** or **repetition**
- Lines are used to show the **flow of control**



## Example

Your notes

**Flowchart**

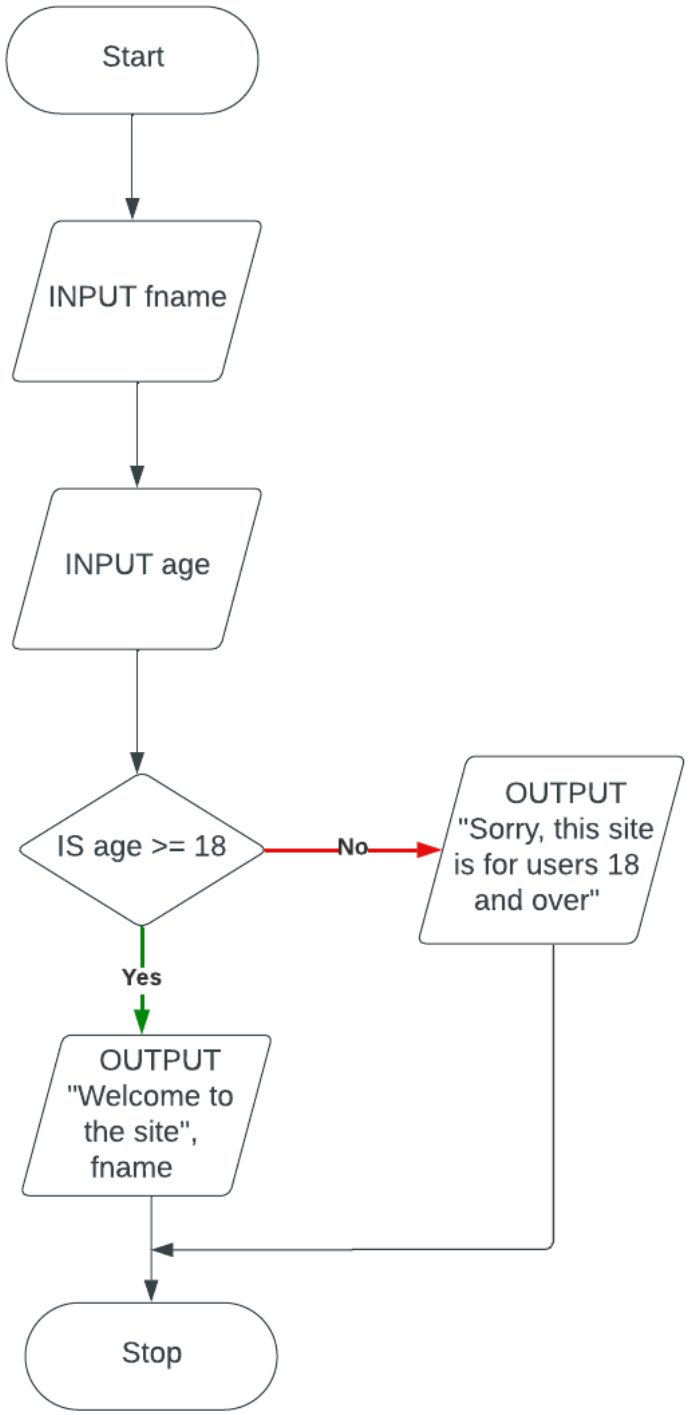


- The casino would like the algorithm **refined** so that the user also **enter their first name** and this is used to **greet the user when they access the site**


Your notes

**Flowchart**

Your notes



Your notes

## Identify Errors in Algorithms


Your notes

- Designing algorithms is a skill that must be developed and when designing algorithms, mistakes will be made
- There are two main types of errors that when designing algorithms a programmer must be able to identify & fix, they are:
  - **Syntax errors**
  - **Logic errors**

# Syntax Errors

## What is a syntax error?

- A syntax error is **an error that breaks the grammatical rules of a programming language** and **stops it from running**
- Examples of syntax errors are:
  - **Typos and spelling errors**
  - **Missing or extra brackets or quotes**
  - **Misplaced or missing semicolons**
  - **Invalid variable or function names**
  - **Incorrect use of operators**
  - **Incorrectly nested loops & blocks of code**

## Examples

| Syntax Errors | Corrected |
|---|---|
| age = input("Enter age)<br><br>favNum == input("Enter favourite number")<br><br>print age + favNum)<br><br>print (age x favNum) | age = input("Enter age") # Missing "<br><br>favNum = input("Enter favourite number")<br># Only one equal sign<br><br>print (age + favNum) # Missing bracket<br><br>print (age * favNum) # Multiply symbol is * |
| num1 = imput("Enter the first number") | num1 = input("Enter the first number") # Misspelt word |

| | |
|---|---|
| num2 = input(Enter the second number)<br>if num1 > num2 then<br>print(num1 + " is larger")<br>elseif num2 > num1 then<br>Print(num2 + " is larger")<br>else<br>print("The numbers are the same")<br>endif | num2 = input("Enter the second number") # Missing quotes<br>if num1 > num2 then<br>print(num1 + " is larger") # Block not indented<br>elseif num2 > num1 then<br>print(num2 + " is larger") # Lowercase p<br>else<br>print("The numbers are the same")<br>endif |


Your notes

# Logic Errors

## What is a logic error?

- A logic error is where **incorrect code is used that causes the program to run**, but **produces an incorrect output or result**
- Logic errors can be difficult to identify by the person who wrote the program, so one method of finding them is to use '**Trace Tables**'
- Examples of logic errors are:
  - **Incorrect use of operators (< and >)**
  - **Logical operator confusion (AND for OR)**
  - **Looping one extra time**
  - **Indexing arrays incorrectly (arrays indexing starts from 0)**
  - **Using variables before they are assigned**
  - **Infinite loops**

## Example

- An algorithm is written to take as input the number of miles travelled. The algorithm works out how much this will cost, with each mile costing £0.30 in petrol. If this is greater than £10.00 then it is reduced by 10%.

| Logic errors | Corrected |
|---|---|

| | |
|---|---|
| miles = input("Enter the number of miles)<br>cost = 0.3<br>if cost = 10 then<br>cost = cost * 0.1<br>endif<br>print(cost) | miles = input("Enter the number of miles")<br>cost = miles * 0.3<br>if cost > 10 then<br>cost = cost * 0.9<br>endif<br>print(cost) |

**Commentary**

- The cost was **set to 0.3** (30p) instead of correctly calculating the cost of the trip by applying the formula, **miles * 0.3**
- The cost should only be reduced if the **cost is greater than 10**, in the original algorithm it only checked if the cost was **equal to 10**
- To calculate a discount of 10%, either **calculate what 10% is and subtract it from the original** or **multiply the full cost by 0.9**. In the original algorithm it calculates what 10% is and sets the cost to equal it.

Your notes

## Worked Example

Nine players take part in a competition, their scores are stored in an array. The array is named scores and the index represents the player

**Array** scores

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Score | 7 | 9 | 2 | 11 | 8 | 4 | 13 | 10 | 5 |

The following program counts the total score of all the players

```
for x = 1 to 8
total = 0
total = total + scores[x]
```

```
next x

print(total)
```

When tested, the program is found to contain **two** logic errors.

Describe how the program can be refined to remove these logic errors **[2]**

**How to answer this question**

- A common logic error if an algorithm contains a loop is checking it loops the correct amount of times, how many times should this algorithm loop?

**Answer**

- For loop changed to **include 0**
- total = 0 moved to **before** loop starts

**Guidance**

- Moving total outside the loop is not enough, it could be moved after the loop which would still be a logic error)
- Corrected code accepted

```
total = 0

for x = 0 to 8

total = total + scores[x]

next x

print(total)
```

Your notes

Your notes

# Trace Tables

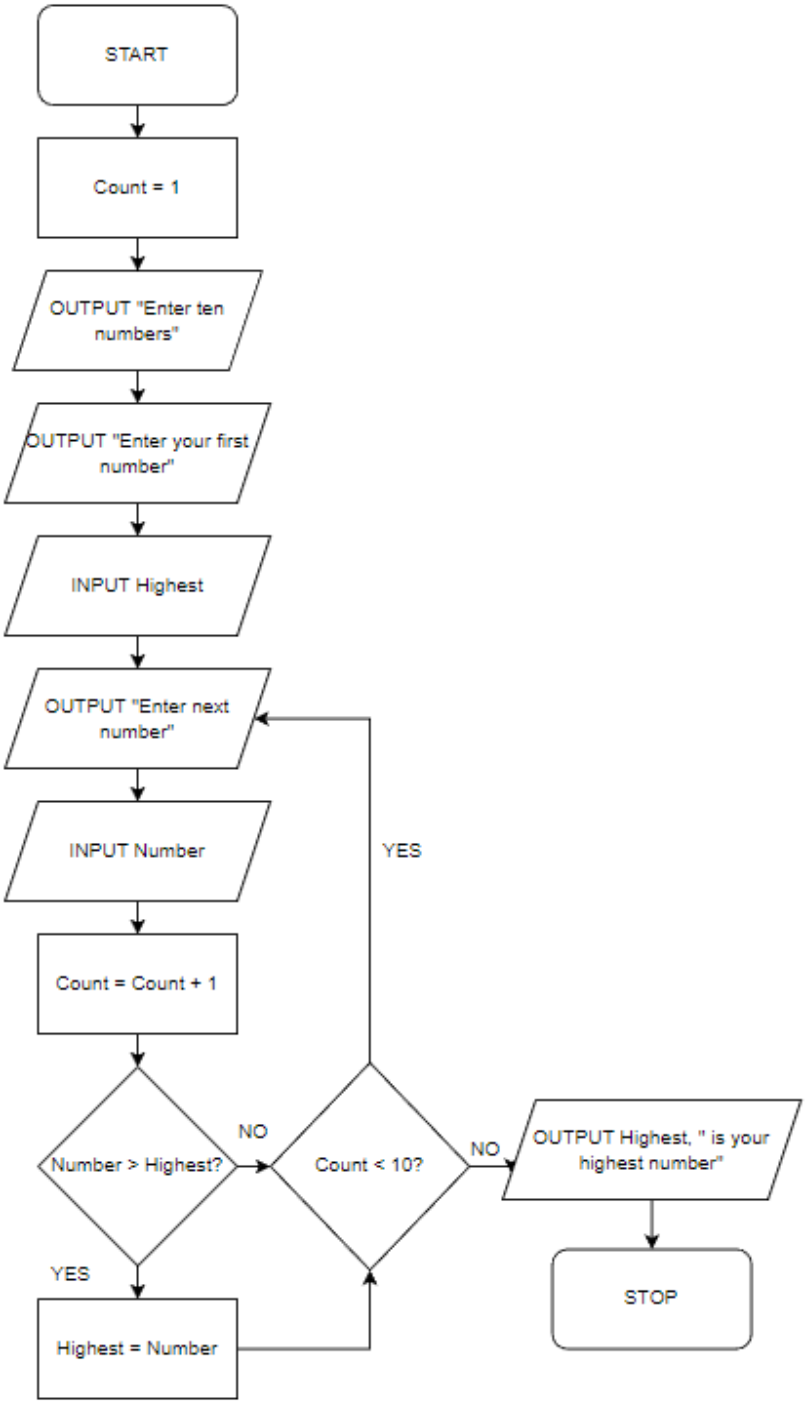
## What is a trace table?

- A trace table is **used to test algorithms and programs for logic errors** that appear when an algorithm or program executes
- Trace tables can be used with **flowcharts, pseudocode** or **program code**
- A trace table can be used to:
  - **Discover the purpose of an algorithm** by showing output data and intermediary steps
  - **Record the state of the algorithm** at each step or iteration
- Each **stage of the algorithm is executed step by step**.
- Inputs, outputs, variables and processes can be **checked for the correct value** when the stage is completed

## Trace table walkthrough

- Below is a flowchart to determine the **highest number of ten user-entered numbers**
- The algorithm prompts the user to enter the first number which automatically becomes the highest number entered
- The user is then prompted to enter nine more numbers.
  - If a new number is higher than an older number then it is replaced
- Once all ten numbers are entered, the algorithm outputs which number was the highest
- Example test data to be used is: **4, 3, 7, 1, 8, 3, 6, 9, 12, 10**

Your notes



| Trace table: Highest number | | | |
|---|---|---|---|
| **Count** | **Highest** | **Number** | **Output** |
| 1 | | | Enter ten numbers |
| | 4 | | Enter your first number |
| 2 | | 3 | Enter your next number |
| 3 | 7 | 7 | |
| 4 | | 1 | |
| 5 | 8 | 8 | |
| 6 | | 3 | |
| 7 | | 6 | |
| 8 | 9 | 9 | |
| 9 | 12 | 12 | |
| 10 | | 10 | 12 is your highest number |

Your notes

## Worked Example

| | |
|---|---|
| **01** | X = 5 |
| **02** | Y = 3 |
| **03** | while X > 0 |
| **04** | Y = Y + 6 |

| | |
|---|---|
| 05 | X = X - 1 |
| 06 | print (Y) |

Your notes

Complete the following trace table for the given algorithm, the first two lines have been filled in for you

| Line number | X | Y | PRINT |
|---|---|---|---|
| 01 | 5 | | |
| 02 | | 3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**Answer**

| Line number | X | Y | PRINT |
|---|---|---|---|
| 01 | 5 | | |
| 02 | | 3 | |
| 04 | | 9 | |
| 05 | 4 | | |
| 04 | | 15 | |
| 05 | 3 | | |
| 04 | | 21 | |
| 05 | 2 | | |
| 04 | | 27 | |
| 05 | 1 | | |
| 04 | | 33 | |
| 05 | 0 | | |
| 06 | | | 33 |