**Chapter 8**

# Introduction

## 2.1.1 Problem-solving and design

| Revision Check List (Based on CAIE Syllabus) | |
|---|---|
| Show understanding that every computer system is made up of sub-systems, which in turn are made up of further sub-systems | |
| Use top-down design, structure diagrams, flowcharts, pseudo code, library routines and subroutines | |
| Work out the purpose of a given algorithm | |
| Explain standard methods of solution | |
| Suggest and apply suitable test data | |
| Understand the need for validation and verification checks to be made on input data (validation could include range checks, length checks, type checks and check digits) | |
| Use trace tables to find the value of variables at each step in an algorithm | |
| Identify errors in given algorithms and suggest ways of removing these errors | |
| Produce an algorithm for a given problem (either in the form of pseudo code or flowchart) | |
| Comment on the effectiveness of a given solution | |

**System** is a set of things working together as parts of a mechanism or an interconnecting network; a complex whole.

**System** is a set of principles or procedures according to which something is done; an organized sc heme or method.

A **system** is a set of rules, an arrangement of things, or a group of related things that work together to perform a function.

A **system** is **made up** of a number of **subsystems**. Each subsystem can be further divided into subsystems and so on until each sub-system just performs a single action.

For example the human body is made up of the circulatory system, the digestive system, the nervous system and so on.

An automobile has an exhaust system, an electrical system, an ignition system and so on.

A **COMPUTER SYSTEM** is made up of hardware, software & data, communications and people; each computer system can be divided up into a set of sub-systems. Each subsystem can be further divided into sub-systems and so on until each sub-system just performs a single action.

Computer system is often divided up into sub-systems. This division can be shown using top-down design to produce structure diagrams that demonstrate the modular construction of the system. Each sub-system can be developed by a programmer as sub-routine or an existing library routine may be already available for use. How each sub-routine works can be shown by using flowcharts or pseudo code.

- Top-down design
- Structure diagrams
- Flowcharts
- Pseudo code
- Library routines
- Sub-routines

**1. Top-Down Design**

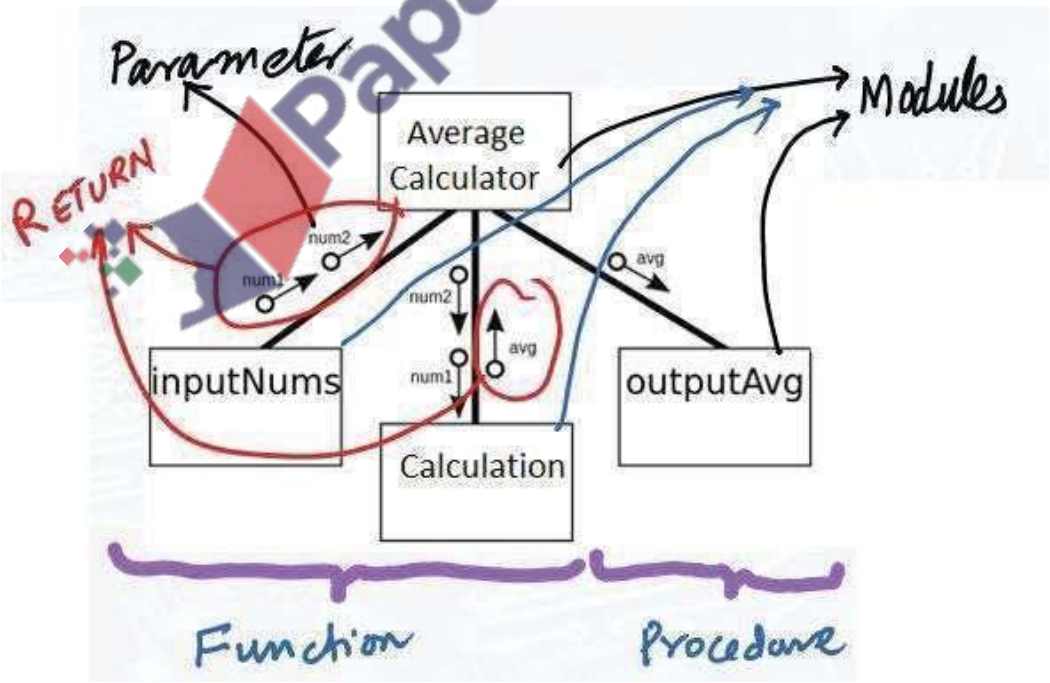
**Top-down design** is the breaking down of a computer system into a set of subsystems, then breaking each sub-system down into a set of smaller sub-systems, until each sub-system just performs a single action.

This is an effective way of designing a computer system to provide a solution to a problem, since each part of the problem is broken down into smaller more manageable problems. The process of breaking down into smaller sub-systems is called 'stepwise refinement'.

This structured approach works for the development of both large and small computer systems. When large computer systems are being developed this means that several programmers can work independently to develop and test different subsystems for the same system at the same time. This reduces the development and testing time.

**2. Structure Diagrams**

The **STRUCTURE DIAGRAM** shows the design of a computer system in a hierarchical way, with each level giving a more detailed breakdown of the system into sub-systems.



**3. Flowcharts**

A **FLOWCHART** shows diagrammatically the steps required for a task (sub-system) and the order that they are to be performed. These steps together with the order are called an **ALGORITHM**. Flowcharts are an effective way to communicate the algorithm that shows how a system or sub-system works.

**4. Pseudo code**

**PSEUDO CODE** is a simple method of showing an algorithm, using English-like words and mathematical operators that are set out to look like a program.

**5. Library routines**

A **LIBRARY ROUTINE** is a set of programming instructions for a given task that is already available for use. It is pre-tested and usually performs a task that is frequently required. For example, the task 'get time' in the checking-for-the-alarm-time algorithm would probably be readily available as a library routine.

**6. Sub-routines**

A **SUB-ROUTINE** is a set of programming instructions for a given task that forms a subsystem, not the whole system. Sub-routines written in high-level programming languages are called 'procedures' or 'functions' depending on how they are used.

**7. Function**

A **Function** is a sub-routine that always returns a value.

**8. Procedure**

A **Procedure** is a sub-routine that doesn't have to returns a value.

**Winter 2018 P22**

**3 Four** programming concepts and **four** descriptions are shown.

Draw a line to connect each programming concept to the most appropriate description. [3]

| Programming concept | Description |
|---|---|
| Library routine | A subroutine that does not have to return a value. |
| Structure diagram | A standard subroutine that is available for immediate use. |
| Procedure | A subroutine that always returns a value. |
| Function | An overview of a program or subroutine. |

Chapter 9

# Algorithm

### 2.1.2 Algorithm Pseudo code

**An algorithm is a series of well-defined steps which gives a procedure for solving a type of problem.**
**The word algorithm comes from the name of 9th century mathematician al-Khwarizmi (Muhammad Bin Musa Al-Khwarizmi).**
**In fact, even the word algebra is derived from his book "Hisab al-jebrw'al-muqabala"**

2.1.2 Pseudo code
• understand and use pseudo code for assignment, using ←
• understand and use pseudo code, using the following conditional statements:
IF … THEN … ELSE … ENDIF
CASE … OF … OTHERWISE … ENDCASE
• understand and use pseudo code, using the following loop structures:
FOR … TO … NEXT
REPEAT … UNTIL
WHILE … DO … ENDWHILE
• understand and use pseudo code, using the following commands and statements:
INPUT and OUTPUT (e.g. READ and PRINT)
totalling (e.g. Sum ← Sum + Number)
counting (e.g. Count ← Count + 1)
(Candidates are advised to try out solutions to a variety of different problems on a computer using a language of their choice; no particular programming language will be assumed in this syllabus.)

"An **algorithm** is a sequence of steps for a computer program to accomplish a task."

In general, an 'algorithm' is the name given to a defined set of steps used to complete a task.

For instance you could define an algorithm to make a cup of tea. You start by filling the kettle, and then place a tea bag in the cup and so on.

In computer terms, an algorithm describes the set of steps needed to carry out a software task.

This mini-web takes you through the topic of algorithm

**Atomic type names**

The following keywords are used to designate atomic data types:

1. **INTEGER:**
   A whole number (without fractional part) like COUNT which never requires fractional part
   For example 56, 89, 1

2. **REAL:**
   A number capable of containing a fractional part like Weight may contain fractional Part
   For example 56.8, 89.0, 1.2

3. **CHAR:**
   A single character (may be letter, special character or number but number cannot be used in calculation )
   For example ‘A’, ‘$’, ‘5’

4. **STRING**:
   A sequence of alphanumeric and special characters but number cannot be used in calculation
   For example “Abdullah”, “0300-2724734”, “House No 56 Block 2, PECHS Karachi”

5. **BOOLEAN:** A data type with two possible values
   For example TRUE and FALSE or YES or NO

6. **DATE:** To store a calendar date
   For example 16/04/2010

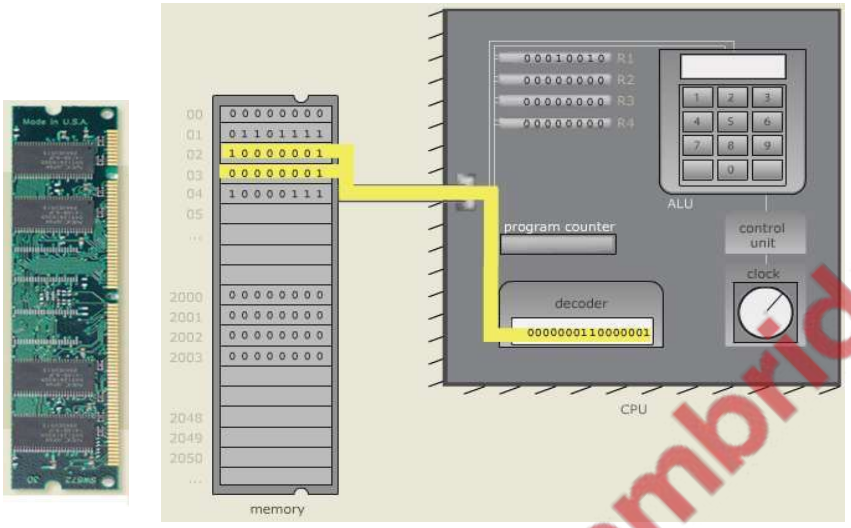
**Literals**

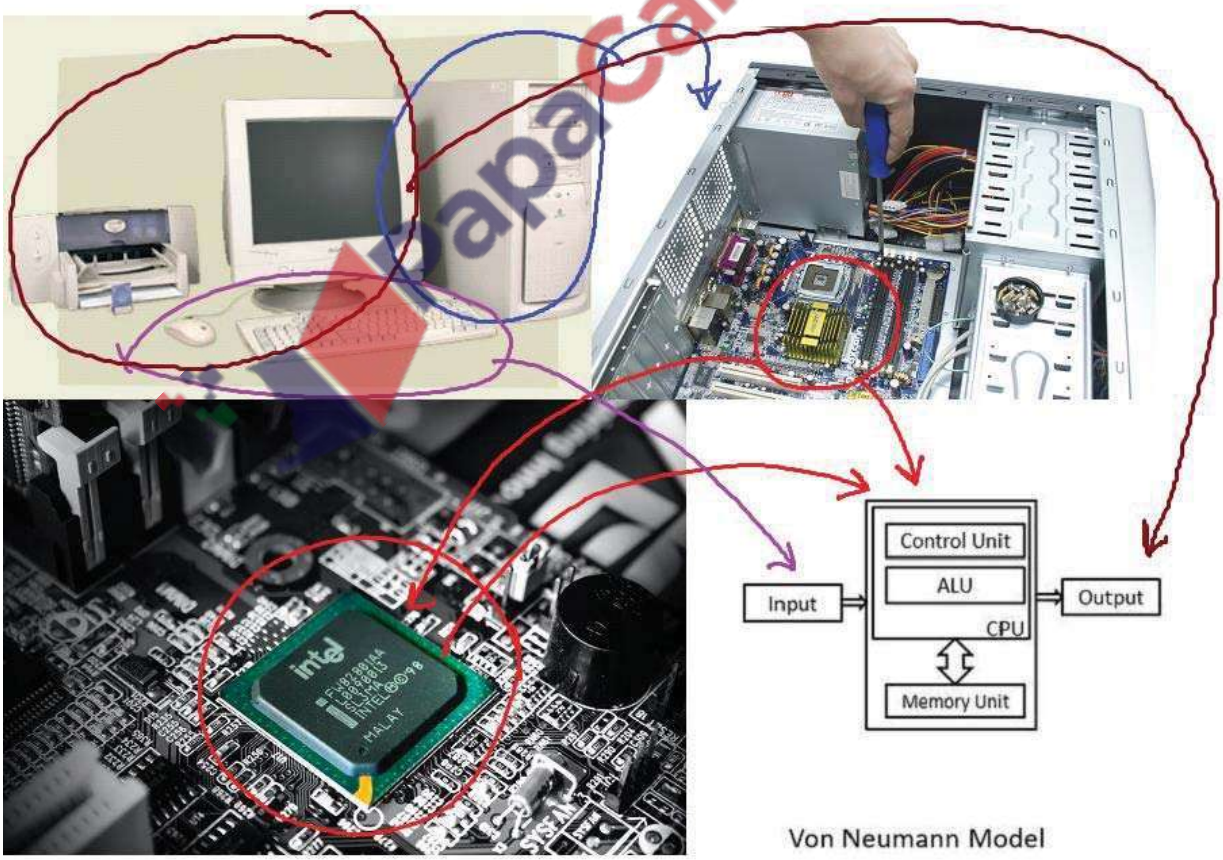
Literals of the above data types are written as follows:

| **Data Type** | **Literals** |
|---|---|
| **Integers**: | Written as normal in the denary system, e.g. 5, -3 |
| **Real**: | Always written with at least one digit on either side of the decimal point, zeros being added if necessary, e.g. 4.7, 0.3, -4.0, 0.0 |
| **Char:** | A single character delimited by single quotes, e.g. 'x', 'C', '@' |
| **String:** | Delimited by double quotes. A string may contain no characters (i.e. the empty string) e.g. "This is a string", "" |
| **Boolean:** | TRUE, FALSE |

**Variable:**
Variable is memory location where a value can be stored. The values stored in a variable are changed during execution.

**Identifiers**
Identifiers (the names given to variables, constants, procedures and functions) are in mix case. They can only contain letters (A–Z, a–z) and digits (0–9). They must start with a letter and not a digit. Accented letters and other characters, including the underscore, should not be used.




As in programming, it is good practice to use identifier names that describe the variable, procedure

or function they refer to. Single letters may be used where these are conventional (such as i and j when dealing with array indices, or X and Y when dealing with coordinates) as these are made clear by the convention.

Cambridge Ordinary Level
2210 Computer Science June 2019
Principal Examiner Report for Teachers

# COMPUTER SCIENCE

**Paper 2210/22**
**Paper 2**

**Key messages**

Candidates must take care when declaring and using variables, constants and arrays as part of a response to ensure that the identifier declared could be used in a program. Identifiers must not contain spaces or other punctuation. Once declared or used the same identifier should be used throughout the answer. Candidates are advised to read through each answer to ensure that no errors have been made.

Cambridge Ordinary Level
2210 Computer Science June 2018
Principal Examiner Report for Teachers

# COMPUTER SCIENCE

**Paper 2210/21**
**Paper 2**

**Key messages**

Candidates who had previously completed the tasks for the pre-release (Computer Shop) were able to demonstrate appropriate techniques for solving this problem using a number of valid interpretations of the tasks. These candidates were able to provide answers for ***Section A*** that demonstrated the programs they had written, descriptions of how they had solved tasks and why they had used their chosen methods.

Candidates who were able to explain their code when requested performed better than those who simply wrote out their code.

Candidates should be careful when answering questions pertaining to a specific task in the pre-release materials that their response is related specifically to that task and not generically to the overall pre-release material, or to programming in general. Also, when declaring variables, constants and arrays, it is important that the identifier declared could be used and would work in a program, i.e. it must follow the rules of the programming language to which it relates. Candidates are further advised to ensure that identifiers are descriptive, rather than vague single characters, to demonstrate good programming practice.

**Keywords** should never be used as variables.

Identifiers should be considered case insensitive, for example, Countdown and Countdown should not be used as separate variables.

**Variable declarations**

It is good practice to declare variables explicitly in pseudo code.

Declarations are made as follows:

```
DECLARE<identifier> : <data type>
```

**Example**

```
DECLARE Surname : STRING
DECLARE FirstName : STRING
DECLARE DateOfBirth : DATE
DECLARE Section : CHAR
DECLARE Counter : INTEGER
DECLARE TotalToPay : REAL
DECLARE GameOver : BOOLEAN
```

**Constant:**

Constant is memory location where a value can be stored but the stored value remaining same during execution.

It is good practice to use constants if this makes the pseudo code more readable, as an identifier is more meaningful in many cases than a literal. It also makes the pseudo code easier to update if the value of the constant changes.

**Constant declaration**

Constants are normally declared at the beginning of a piece of pseudo code (unless it is desirable to restrict the scope of the constant).

Constants are declared by stating the identifier and the literal value in the following format:

```
CONSTANT<identifier> = <value>
```

**Example**

```
CONSTANT HourlyRate = 6.50
CONSTANT DefaultText = “N/A”
```

Only literals can be used as the value of a constant. A variable, another constant or an expression must never be used.

**Input and output**

Values are input using the INPUT command as follows:

```
INPUT <identifier>
```

The identifier should be a variable (that may be an individual element of a data structure such as an array, or a custom data type).

Values are output using the OUTPUT command as follows:

```
OUTPUT <value(s)>
```

Several values, separated by commas, can be output using the same command.

**Example – INPUT and OUTPUT statements**

```
INPUT Answer
OUTPUT Score
OUTPUT "You have ", Lives, " lives left"
```

**Note** that the syllabus for IGCSE (0478) gives READ and PRINT as examples for INPUT and OUTPUT, respectively.

**Arithmetic operations**

Standard arithmetic operator symbols are used:

- + Addition
- - Subtraction
- * Multiplication
- / Division

Care should be taken with the division operation: the resulting value should be of data type REAL, even if the operands are integers.

The integer division operators MOD and DIV can be used. However, their use should be explained explicitly and not assumed.

Multiplication and division have higher precedence over addition and subtraction (this is the normal mathematical convention). However, it is good practice to make the order of operations in complex expressions explicit by using parentheses.

**Logic operators**

The only logic operators (also called relational operators) used are AND, OR and NOT. The operands and results of these operations are always of data type BOOLEAN.

In complex expressions it is advisable to use parentheses to make the order of operations explicit.

**Comments**

Comments are preceded by two forward slashes // . The comment continues until the end of the line. For multi-line comments, each line is preceded by //.

Normally the comment is on a separate line before, and at the same level of indentation as, the code it refers to. Occasionally, however, a short comment that refers to a single line may be at the end of the line to which it refers.

**Example – comments**

```
// This is example of comments
// swapping values of X and Y
Temp ← X // temporarily store X
X ← Y
Y ← Temp
```

**COUNTING**

Counting is used to find how many items are there by incrementing by 1 during each time loop is executed.

It is sometimes necessary to count how many times something happens.

To count up or increment by 1, we can use statements such as:

Count ← Count + 1

(new) (old)

i.e. INCREMENT (old) Count by 1 to get (new) Count

**TOTALLING**

Totalling is used to calculate running total. We can use a variable such as Total or Sum to hold the running total and assignment statements such as:

Total ← Total + Number

(new) (old)

i.e. ADD Number to (old) Total to obtain (new) Total

**Q 1 Summer 2015 P21& 23**

**5** Explain the difference between a variable and a constant in a program.

.....

.....

.....

.....[2]

*Examiner Report Question 5*
*Well answered by many candidates.*

**Q 2 Summer 2015 P21& 23**

**4** Five data types and five data samples are shown below.

Draw a line to link each data type to the correct data sample. [4]

| Data type | Data sample |
|---|---|
| Integer | 'a' |
| Real | 2 |
| Char | 2.0 |
| String | True |
| Boolean | "Twelve" |

*Examiner Report Question 4*

*Nearly all candidates could link the data type of Boolean with the correct data sample. Some candidates confused Real and Integer data types and/or String and Char data types.*

**Summer 2016 P21 &P23**

**3** A program will be written to store information about members of a swimming club.

The following membership details will be recorded:

- Name
- Gender
- Status:
  - Senior
  - Junior
- Fee
- Team member (Yes or No)

**(i)** Choose a suitable data type for each of the membership details to be recorded. [5]

| **Membership details** | **Data type** |
|---|---|
| Name | |
| Gender | |
| Status | |
| Fee | |
| Team member | |

**Q 4 A** marathon runner records their time for a race in hours, minutes and seconds.

An algorithm is shown below in structured English.

INPUT race time as hours, minutes and seconds

CALCULATE race time in seconds

STORE race time in seconds

OUTPUT race time in seconds

The identifier table needs to show the variables required to write a program for this algorithm. Complete the table.

| **Identifier** | **Data type** | **Description** |
|---|---|---|
| RaceHours | INTEGER | The hours part of the race time. |
| | | |
| | | |
| | | |

**Q 5** A program contains the following code to calculate the circumference of a bicycle wheel, using the wheel size (diameter).

```
CONSTANT Pi = 3.14
INPUT WheelSize
Circumference = Pi * WheelSize
OUTPUT Circumference
```

**(a)** The code uses one constant and two variables.

**(i)** State the names of the constant and the variables.

Constant: .....

Variables: .....[2]

**(ii)** Explain **one** difference between a constant and a variable.

.....

.....

.....[2]

**(b)** The data type of WheelSize is integer and the data type of Circumference is real number.

Explain the difference between an integer and a real number.

.....

.....

.....

..... [2]

**Q 6** Computer programs have to evaluate expressions.

Study the sequence of pseudo code statements.

Write down the value assigned to each variable.

| | |
|---|---|
| DECLARE h, z, w, r, Perimeter, Area: REAL<br>DECLARE A: BOOLEAN<br>h ← 13.6<br>w ← 6.4<br>Perimeter ← (h + w) * 2 | Perimeter =…….. (1) |
| r← 10<br>Area ← 3.14 * (r ^ 2) | Area= (1) |
| z ← 11 + r / 5 + 3 | Z = (1) |
| A ← NOT (r > 10) | A = (1) |

**Q 7** Computer programs have to evaluate expressions.

Study the sequence of pseudo code statements.

Give the value assigned to each variable.

The statement may generate an error. If so, write ERROR.

The & operator is used to concatenate strings.

| | |
|---|---|
| DECLARE N1 : INTEGER | |
| DECLARE N2 : INTEGER | |
| DECLARE Answer : REAL | |
| DECLARE Found : BOOLEAN | |
| DECLARE IsValid : BOOLEAN | |
| N1 ← 3 | |
| N2 ← 9 | |
| Answer ← (N1 + N2) / 6 | Answer = [1] |
| Answer ← 3 * (N1 – 2) + N2 / 2 | Answer = [1] |
| IsValid← (N1 > N2) AND (N2 = 9) | IsValid = [1] |
| Found ← FALSE | |
| IsValid← (N1 > N2 / 2) OR (Found = FALSE) | IsValid = [1] |
| Answer ← "1034" & " + " & "65" | Answer = [1] |

**Q 8 March 2017 P21 (India)**

**3** There is a program that stores the following data: [8]

• EmployeeID, an employee ID which must be two letters followed by 4 numbers, e.g. TY4587

• Manager, whether the employee is a manager or not

• AnnualHoliday, number of whole days' annual holiday

• PayGrade, the employee's pay grade which must be a single letter A–F

Complete the following table to identify:

• The most appropriate data type for each variable

| **Variable** | **Data type** |
|---|---|
| EmployeeID | |
| Manager | |
| AnnualHoliday | |
| PayGrade | |

**2** Describe, giving an example for each, the following data types used in programming.

**Integer** Description .....

.....

Example .....

**String** Description .....

.....

Example ..... [4]

**Q10 (i)** Programming languages support different data types.

Complete the table by giving a suitable data type for each example value. [4]

| Example value | Data type |
|---|---|
| 43 | |
| TRUE | |
| - 273.16 | |
| "- 273.16" | |

**(ii)** Programming languages support different data types.

Complete the table by giving a suitable data type for each example value. [4]

| Example value | Data type |
|---|---|
| "NOT TRUE" | |
| - 4.5 | |
| NOT FALSE | |
| 132 | |

**(b)** Program variables have values as follows:

Programming languages support different data types.

Give an appropriate data type for the following variables from **part (b)**. [5]

| Variable | Value | Data type |
|---|---|---|
| Married | 03/04/1982 | |
| ID | "M1234" | |
| MiddleInitial | 'J' | |
| Height | 5.6 | |
| IsMarried | TRUE | |
| Children | 2 | |

**2** Describe each of the following data types used in programming. In each case, give an example of a piece of data to illustrate your answer. Each example must be different.

Char.....

.....

.....

String.....

.....

.....

Integer .....

.....

.....

Real .....

.....

.....

Date .....

.....

.....

Boolean.....

.....

.....[12]

**Q 12 Winter 2019 P22**

**6** Explain why constants, variables and arrays are used in programming.

Constants .....

.....

.....

.....

Variables .....

.....

.....

..... [2]

**The concept of a program**

A program is a sequence of instructions or programming language statements written to make a computer perform certain tasks.

**Basic Control Constructs:**

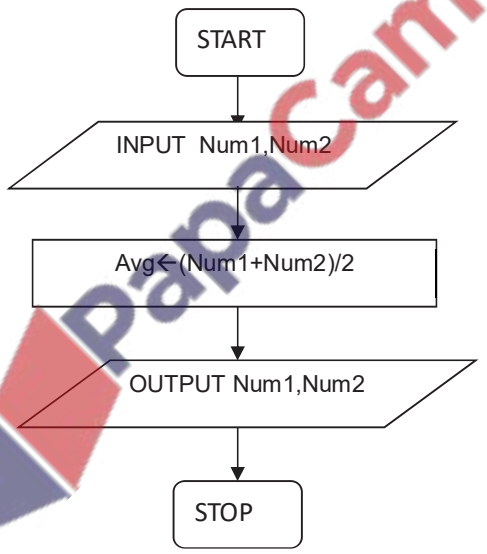
Following are the basic constructs of algorithm and program which controls execution of statements:

1. **Sequence:** One statement is being executed after another in the order they are written
   In following example statement number 'i' will be executed at 1st and then 'ii' then 'iii' and at last statement number 'iv' will be executed:
   i. INPUT Num1
   ii. INPUT Num2
   iii. Total ← Num1 + Num2
   iv. PRINT Total

Flowchart is also drawn in the sequence in which the program is intended to be executed.

Write an algorithm, using flowchart only, which:

- Inputs two numbers
- Calculate their average
- Output average




Problem 1: Input two numbers and output their sum

Problem 2: Input daily wages and number of day worked and output monthly pay.

Q 9.1) Describe the term Computer System and name it's components.

.....

.....

.....

.....

.....

..... [5]

Q 9.2 a) Define the term **algorithm,** name the two ways of representing algorithm.

.....

.....

..... [1]

1. .....

2. ..... [2]

***Answer Key:*** *A series of instructions//sequence of steps;(Designed to) perform a particular task//solve a problem.*

*Flowchart and pseudo code*

b) Simple algorithms usually consist of three different stages.
Complete the table below. Write each example statement in **program code**.
The second stage has already been given. [5]

| **Stage** | **Example statement** |
|---|---|
| | |
| Process | |
| | |

Q 9.3) What is top-down design

.....

.....

..... [1]

Q 9.4) Describe following terms and give one example of each

1. Library Routine

.....

.....

..... [1]

2. Sub-routine

.....

.....

..... [1]

3. Function

.....

.....

..... [1]

4. Procedure

.....

.....

..... [1]

5. Rogue Value

.....

.....

..... [1]

2. **Assignment:** Storing values in a variable is known as assignment.
   The assignment operator is ←.
   Assignments should be made in the following format:
   <identifier> ← <value>

| <identifier> ← <value> |
|---|

| <identifier> ← <expression> |
|---|

**<u>For example:</u>**
Counter ← 0
Counter ← Counter + 1
TotalToPay ← NumberOfHours * HourlyRate

| Counter ← 0 |
|---|

| Counter ← Counter + 1 |
|---|

| TotalToPay ← NumberOfHours * HourlyRate |
|---|

3. **Selection (Condition):** Selection determines program flow path on the basis of given condition.

It also decides which statement(s) are to be executed depending upon the result of a given condition. In the following example statement number 'I' will be executed at 1st and then number 'ii'. Execution of statement number 'iii' and 'v' depends upon the result of condition given condition in statement number 'ii':

```
i.   INPUT Marks
ii.  IF Marks>= 50 THEN
iii.         PRINT "Pass"
iv.  ELSE
v.           PRINT "Fail"
vi.  ENDIF
```

4. **Iteration (Loop or Repetition):** Iteration is used to execute a set of instructions multiple times. It is also referred as LOOP or ITERATION.

   In the following example statement number 'ii' will be executed 10 times:

```
i.   FOR Count ← 1 TO 10
ii.        PRINT "Allah is the only God"
iii. NEXT Count
```

A computer's processor can only run a computer program in the form of a file of machine code, which is a sequence of binary codes representing instructions for the processor.

The instruction set for a family of processors is the machine language in which machine code is written for that family of processors.

When machine code runs, the processor repeatedly:

- Fetches an instruction from internal memory
- Decodes the instruction
- Executes the instruction.

**Selection:**

Selection determines program flow path on the basis of given condition.

Selection decides which statement(s) are to be executed depending upon the result of a given condition.

For selection following statements are used:

- IF
- CASE

**IF statements**

IF statements are used when there are one or two options.

When there is only one option IF statements without an ELSE clause is written as follows:

```
IF<condition>THEN
        <statements if true>
ENDIF
```

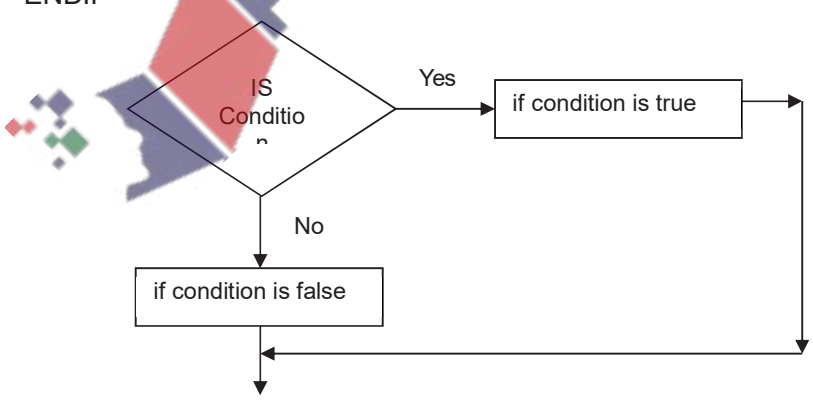
**Example**

```
IF Number>Largest THEN
    Largest←Number

ENDIF
```

When there are two options IF statements with an ELSE clause is written as follows:

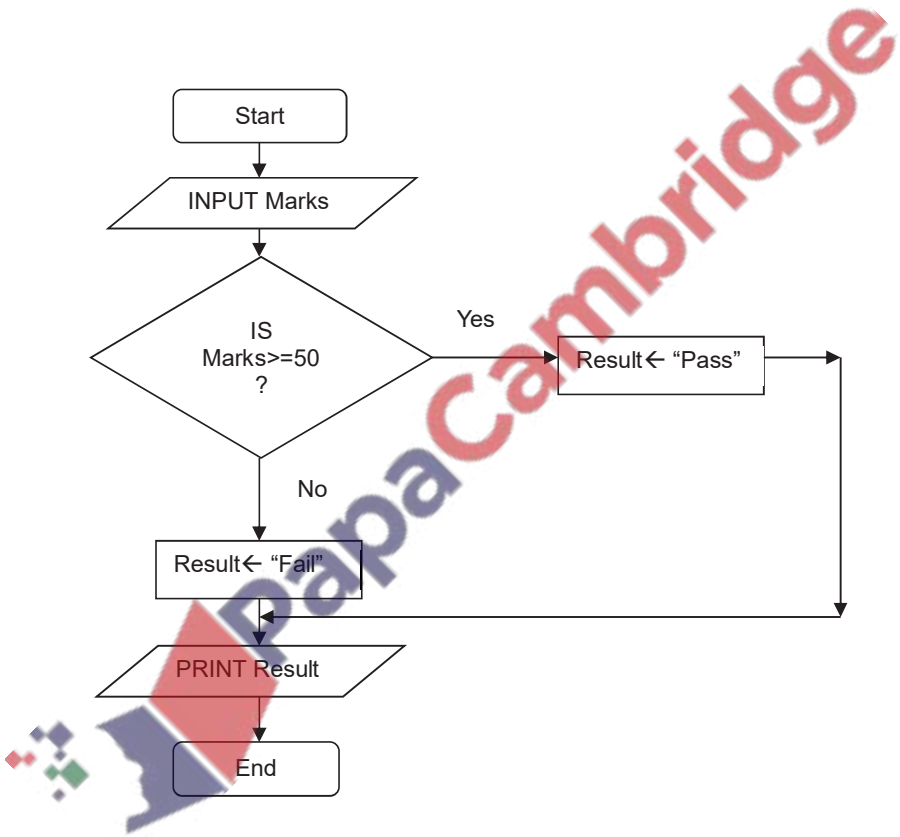
```
IF <condition>THEN
        <statements if true>
  ELSE
        <statements if false>
ENDIF
```



**Example**

```
IF Marks>=50 THEN
        Result ← "Pass"
  ELSE
        Result ← "Fail"
ENDIF
PRINT Result
```

Note that the THEN and ELSE clauses are only indented by two spaces. (They are, in a sense, a continuation of the IF statement rather than separate statements).
When IF statements are nested, the nesting should continue the indentation of two spaces. In particular, run-on THENIF and ELSE IF lines should be avoided.



**CASE statements**

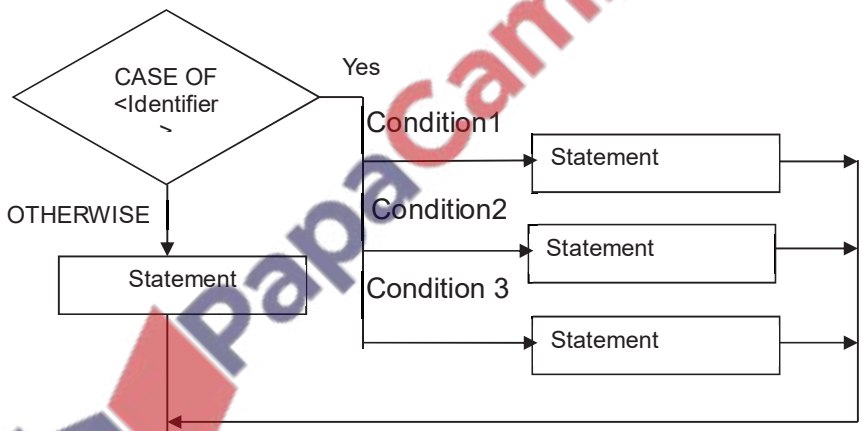
CASE is a conditional statement to deal with many possible outcomes.

CASE statements allow one out of several branches of code to be executed, depending on the value of a variable.
In case selection number of statements are reduced so code become more simplified.
CASE statements are written as follows:

```
CASE OF<identifier>
<value 1> : <statement>
<value 2> : <statement>
  ...
ENDCASE
```

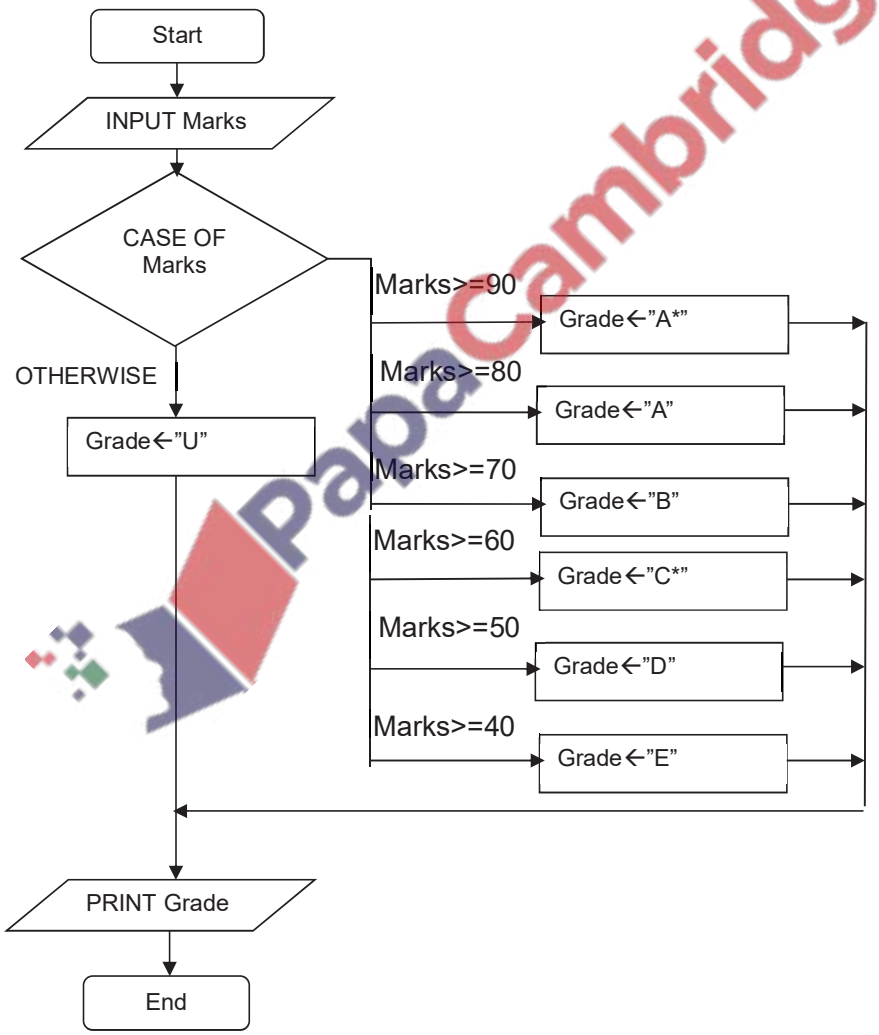
An OTHERWISE clause can be the last case:

```
CASE OF <identifier>
<value 1> : <statement>
<value 2> : <statement>
  ...
  OTHERWISE<statement>
ENDCASE
```



**Example – formatted CASE statement**

```
INPUT Marks
CASE Marks OF
>=90: Grade ← "A*"
>=80: Grade ← "A"
>=70: Grade ← "B"
>=60: Grade ← "C"
>=50: Grade ← "D"
>=40: Grade ← "E"
OTHERWISE : : Grade ← "U"
ENDCASE
PRINT Grade
```



| IF…THEN…ELSE…ENDIF | CASE….OF….OTHERWISE…ENDCASE |
|---|---|
| | |
| Problem: input marks and output result | Problem: input marks and output grade |

| Problem: input marks and output grade | |
|---|---|
| IF | CASE |
| | |

Problem 3: Input marks and output Result, the passing marks is 40 or above.

HW Write algorithm using pseudo code for the following problems:

Problem 4: Input total weight of passengers in a lift output "Overloaded, step out" if Total Weight is above 600 otherwise output "Ready to go".

Problem 5: Input age of candidates for driving license, output "Not allowed to drive" or "Kindly fill in the form". The minimum allowed age for driving is 18 years.

Problem 6: Input age of candidate in an employment center, output "You are not eligible due to age". Allowed age is between 18 and 60 both inclusive.

Problem 7: which inputs price and quantity calculates amount and if billing amount is above 5000 then allows a 5% discount on the billing amount.
Output billing amount, discount and amount after discount

**Problem 8) Summer 2018 P22**

**3** This pseudo code algorithm inputs two non-zero numbers and a sign, and then performs the calculation shown by the sign. An input of zero for the first number terminates the process.

```
INPUT Number1, Number2, Sign

IF Sign = '+' THEN Answer = Number1 + Number2 ENDIF
IF Sign = '-' THEN Answer = Number1 - Number2 ENDIF
IF Sign = '*' THEN Answer = Number1 * Number2 ENDIF
IF Sign = '/' THEN Answer = Number1 / Number2 ENDIF

IF Answer <> 0 THEN OUTPUT Answer ENDIF
```

**(a)** Complete the trace table for the input data:
5, 7, +, 6, 2, –, 4, 3, *, 7, 8, ?, 0, 0, / [3]

| Number1 | Number2 | Sign | Answer | OUTPUT |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

(b) Show how you could improve the algorithm written in pseudo code by writing an alternative type of conditional statement in pseudo code. [3]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

**Problem 9) March 2018 P22 (India)**

**5** Explain the difference between the programming concepts of **sequence** and **selection**. Include an example of a programming statement for each concept in your explanation. [4]

……

……

……

……

……

……

……

……

……

***Comments on Question 5***

*Candidates found the explanation of the difference between the programming concepts **sequence** and **selection** challenging, with few candidates identifying that programming statements in a sequence were executed one after another whilst selection meant that the path through the program depends on the result of a question. Candidates were more successful in providing suitable examples of programming statements.*

*Common errors included confusing sequence or selection with iteration.*

**Problem 10) Winter 2018 P22**

**4** A programmer wants to test that the readings from 2000 electricity meters are greater than 400 units and less than 900 units. The programmer uses selection and repetition statements as part of the program. Explain, using programming statements, how selection and repetition could be used in this program.

[2]

……

……

……

……

……

……

**Problem 11) Winter 2018 P23**

**3** Give an example of a pseudo code statement or statements to perform each of the following functions.

A conditional statement [3]

……

……

……

……

**Problem 12) Winter 2015 P21 & 22**

**5** Identify **two** different conditional statements that you can use when writing pseudo code.

1 …….

2 …… [2]

*Examiners' Comments Question 5*

*Many candidates could identify IF as a conditional statement. Candidates with stronger responses throughout also identified CASE.*

**Problem 13) Summer 2016 P22**

6 Identify two different selection statements that you can use when writing pseudo code.

1

2 [2]

**Problem 14) Winter 2016 P22**

**4** IF ... THEN ... ELSE ... ENDIF and CASE ... OF ... OTHERWISE ... ENDCASE

are two different conditional statements that you can use when writing pseudo code.

Explain, using examples, why you would choose to use each conditional statement.

Example 1

Reason for choice

Example 2

Reason for choice [6]

**Problem 15) Winter 2017 P22**

**4** IF … THEN … ELSE … ENDIF is one type of conditional statement used when writing pseudo code.

Identify and describe **another** type of conditional statement that you could use when writing pseudo code. Give a reason why you would use this type of conditional statement.

Conditional statement

Description

Reason [4]

**Problem 16) Summer 2018 P21**

**5** Explain the difference between the programming concepts of **counting** and **totalling**.
Include an example of a programming statement for each concept in your explanation. [4]

**Problem 17) Summer2019 P21**

**3 (a)** Give an example of a conditional statement using pseudo code. [2]

**(b)** Describe the purpose of a conditional statement. [2]

**Problem 18) Winter 2019 P23**

**4** The following pseudocode algorithm uses nested IF statements.

```
IF Response = 1    THEN
        X ← X + Y
    ELSE
        IF Response = 2  THEN
                X=X-Y
            ELSE
                IF Response =3  THEN
                        X= X * Y
                        IF Response = 4   THEN
                                X = X/Y
                            ELSE
                                OUTPUT "No response"
                        ENDIF
                ENDIF
        ENDIF
ENDIF
```

**(a)** Name the type of statement demonstrated by the use of IF … THEN … ELSE … ENDIF [1]

**(b)** Re-write the pseudo code algorithm using a CASE statement. [4]

**Problem 19 (from AS)**

The following pseudocode algorithm has been developed to check whether a string contains a valid password.

To be a valid password, a string must:

- be longer than 6 characters
- contain at least one lower case letter
- contain at least one upper case letter
- contain at least one non-alphabetic character.

```
10 FUNCTION Check(InString : STRING) RETURNS BOOLEAN
11
12      DECLARE Index, StrLen, NumUpper, NumLower, NumNonAlpha : INTEGER
16      DECLARE NextChar : CHAR
18      NumUpper ← 0
19      NumLower ← 0
20      NumNonAlpha ← 0
22      StrLen ← LENGTH(InString)
23      IF StrLen < 7
24        THEN
25              RETURN FALSE
26       ELSE
27              FOR Index← 1 TO StrLen
28                      NextChar ← MID(InString, Index, 1)
29                      IF NextChar >= 'a' AND NextChar <= 'z'
30                           THEN
31                                NumLower ←  NumLower + 1
32                      ELSE
33                                IF NextChar > 'A' AND NextChar <= 'Z'
34                                     THEN
35                                          NumUpper ← NumUpper + 1
36                                ELSE
37                                          NumNonAlpha ← NumNonAlpha + 1
38                                ENDIF
39                      ENDIF
40              ENDFOR
41      ENDIF
43      IF (NumUpper >= 1) AND (NumLower >= 1) AND (NumNonAlpha >= 1)
44          THEN
45              RETURN TRUE
46      ELSE
47              RETURN FALSE
48      ENDIF50 ENDFUNCTION
```

Rewrite lines 29 to 39 of the original pseudocode using a CASE structure. [4]

**Q13 Summer 2019 AS P21**

**2 (c)** The following lines of code are taken from a program in a high-level language.

```
ON x {
        15: Call ProcA
        20: y := 0
        25: y := 99
        NONE: Call ProcError
        }
```

Identify the type of control structure **and** describe the function of the code.

Control structure .....

Description .....

.....

.....

.....[3]

**Exercise on Selection**

**Q1**a) Using pseudo code or otherwise, write an algorithm which will input any three numbers and then print the smallest number.

b) Write an algorithm to input three different numbers, and then output the largest number. Use either pseudo code or a flowchart.

Q 2) Write an algorithm, using pseudo code, to input three different numbers, multiply the two larger numbers together and output the result. Use the variables: Number1, Number2 andNumber3 for your numbers and Answer for your result.

Q 4) Write an algorithm, using pseudo code, to input a number between 0 and 100 inclusive. The algorithm should prompt for the input and output an error message if the number is outside this range.

Q 8) Input price and quantity, calculates amount and if billing amount is above 5000 then allows a 5% discount on the billing amount.
Output billing amount, discount and amount after discount

Q 9) Employees of a shop are entitled to a discount of 10% on the value of goods bought from the shop. However if an employee has worked at the shop for five or more years they are entitled to a discount of 20%. Only employees are allowed discounts. The discount on electrical goods is fixed at only 10%.
Using pseudo code or otherwise, write an algorithm which will determine what discount applies when any person buys an item.

Q 10) Customers can withdraw cash from an Automatic Teller Machine (ATM).

• withdrawal is refused if amount entered > current balance

• withdrawal is refused if amount entered > daily limit

• if current balance < $100, then a charge of 2% is made

• if current balance $100, no charge is made

Write an algorithm which inputs a request for a sum of money, decides if a withdrawal can be made and calculates any charges. Appropriate output messages should be included.

Sample algorithm:

```
input amount
    if amount > balance then x = 1                          (2 marks)
        else if amount > daily limit then x = 1             (1 mark)
        else x = 0
        while x = 0
            if balance < 100 then charge = 0.02 * amount    (1 mark)
                else charge = 0
                                                            (1 mark)
        endwhile
if x = 1 then print "Sorry, withdrawal refused"
print charge                                                (1 mark)
```

**Marking points**

1 mark for checking if amount > balance
1 mark for checking if amount > daily limit
1 mark for some way of testing if withdrawal will be refused (value of x in above)
1 mark for checking if balance < $100…
1 mark …for calculating 2% charge
1 mark for no charge if balance >= $100
2 marks for giving correct outputs [5]

**Q 11)** A formula for calculating the body mass index (BMI) is:

$$\text{BMI} = \frac{\text{weight in kilograms}}{(\text{height in metres}) \times (\text{height in metres})}$$

Using **Flowchart**, write an algorithm that will input weight (kg) and height (m) of students, calculate their body mass index (BMI) and output their BMI and comments on BMI.

| | |
|---|---|
| BMI <19 | Under weight |
| BMI < =25 | Normal Weight |
| BMI>25 | Over weight |

**Q12)** A system uses 5 digit numbers with an additional sixth digit used as a check digit.
(b) Each of the six digits in the number has a digit position. [Total=6]

| | | | | | | |
|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | 1 | ←Digit position |
| a | b | c | d | e | f | |
| | | | | | Check digit | |

digit in position 1 is the check digit i.e. f
The validity of the check digit is found using the following calculation:

- multiply each digit by its digit position (i.e. ax6, bx5, so on)
- add together the results of the multiplications
- divide the sum by 11
- If the remainder is ZERO then the number is valid

**Write an algorithm, using flowchart only, which**

- inputs six-digit barcodes in the form a, b, c, d, e and f
- re-calculates the check digit for each number and checks whether the input check digit (e) is correct

**Q 13) Summer 2013**

A small shop uses barcodes which represent 5 digits. The last digit is used as a check digit.
For example:
a b c d e
0 1 2 3 4
The check digit (e) is found by:
• multiplying the first and third digits (i.e. a and c) by 3
• multiplying the second and fourth digits (i.e. b and d) by 2
• adding these four results together to give a total
• dividing this total by 10
• remainder is check digit (e)

**Write an algorithm, using flowchart only, which**

- inputs five-digit barcodes in the form a, b, c, d, e
- re-calculates the check digit for each number and checks whether the input check digit (e) is correct

**Iteration (Repetition, Loop)**

Repetition is used to execute a set of instructions multiple times.
Repetition is also referred as LOOP or ITERATION.
There are following three types of loops:

1. Count-controlled loop
2. Pre-condition loop
3. Post-condition loop

**Count-controlled (FOR) loops**

Count-controlled loop is used when the number of repetition is already known.
Count-controlled loops are written as follows:

```
FOR <identifier>←<value1> TO <value2>
        <statements>
NEXT<identifier>
```

The identifier must be a variable of data type INTEGER, and the values should be expressions that evaluate to integers.

It is good practice to repeat the identifier after NEXT.

```
FOR <identifier>←<value1> TO <value2> STEP<increment>
        <statements>
NEXT
```

The increment must be an expression that evaluates to an integer. In this case the identifier will be assigned the values from value1 in successive increments of increment until it reaches value2. If it goes past value2, the loop terminates. The increment can be negative.

**Example: to input 10 numbers and output their final total**

```
Total ← 0
FOR Count ← 1 TO 10
INPUT Number
   Total ←Total + Number
NEXT Count
OUTPUT "The grand total is ", Total
```

**Example: to print 1st 10 even numbers**

```
FOR Count ← 1 TO 20STEP 2
      PRINT Count
NEXT Count
```

**Pre-condition (WHILE) loops**

A loop in which condition is given at the start of loop and which is executed only when the condition is true, is called pre-condition loop.

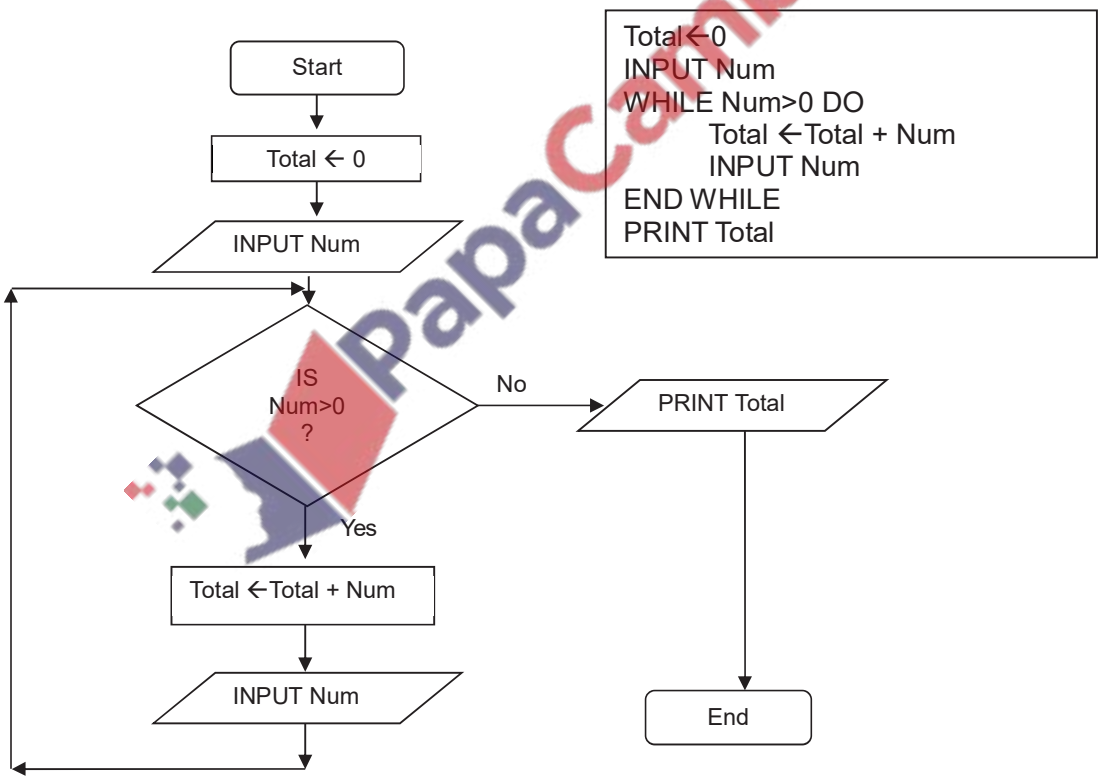
Pre-condition loops are written as follows:

```
WHILE<condition to repeat> DO
        <statements>
ENDWHILE
```

The condition must be an expression that evaluates to a Boolean.
The condition is tested before the statements, and the statements will only be executed if the condition evaluates to TRUE. After the statements have been executed the condition is tested again.
The loop terminates when the condition evaluates to FALSE.
The statements will not be executed if, on the first test, the condition evaluates to FALSE.

**Example: To input a series of numbers and calculate total and stops if a –ve number is entered:**

The condition is checked at the beginning of the loop. If condition is true loop statements are executed again and again.



**Post-condition (REPEAT UNTIL) loops**

A loop in which condition is given at the end of loop and which is executed only when the condition is false is called post-condition loop.
It is are written as follows:

```
REPEAT
        <Statements>
UNTIL<condition to stop the loop>
```

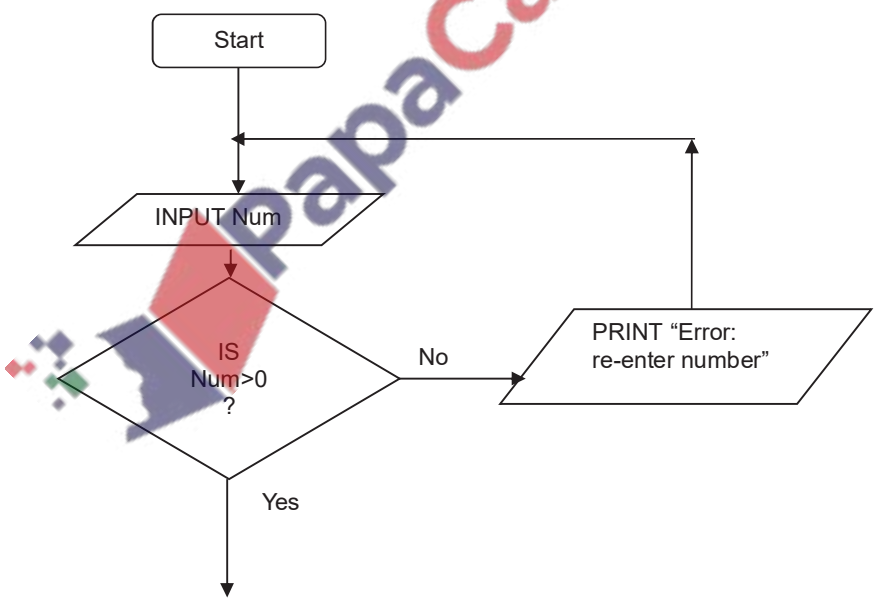
The condition must be an expression that evaluates to a Boolean.

The statements in the loop will be executed at least once. The condition is tested after the statements are executed and if it evaluates to TRUE the loop terminates, otherwise the statements are executed again.

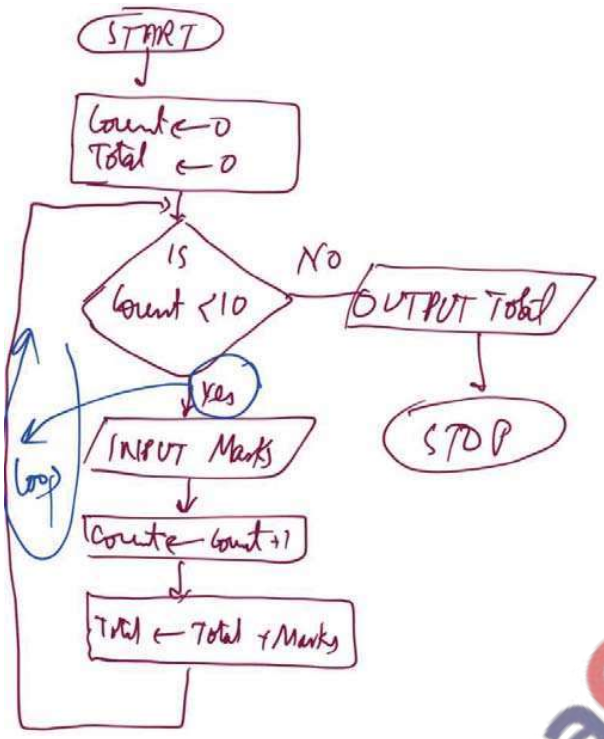
**Example: To input and validate a number and to reject it if a negative number is entered and ask to re-enter another number**

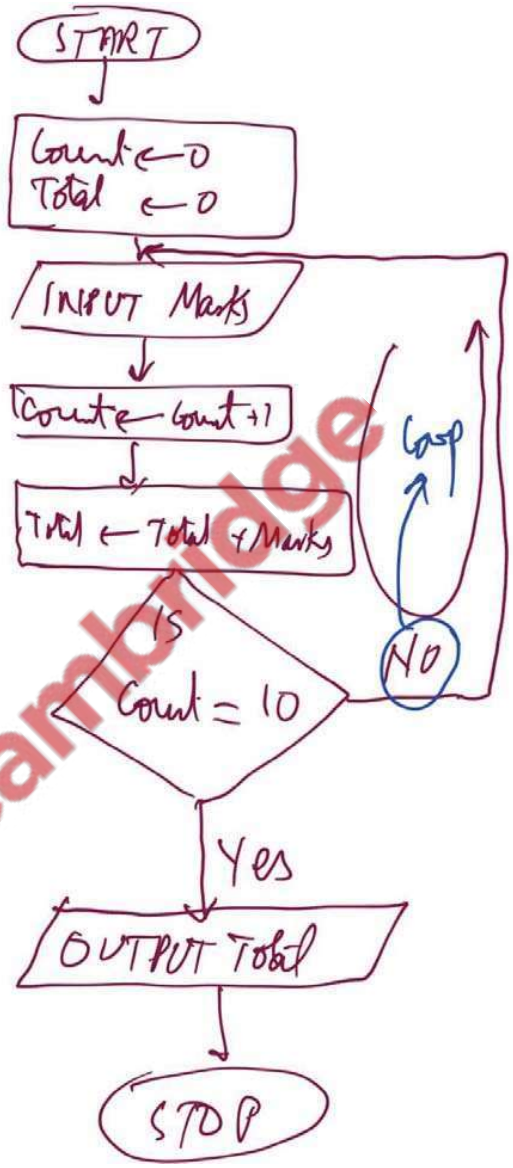
The condition is checked at the end of the loop. If condition is false loop statements are executed again and again.

```
REPEAT
        INPUT Num
UNTIL Num>0
```



**Example: to input 10 numbers and output their final total**



**Control Construct: Iteration:** Iteration is used to execute a set of instructions multiple times. It is also referred as LOOP or ITERATION.

In the following example statement number 'ii' will be executed 10 times:
Problem: Print the name of Allah 10 times.

LOOPING STATEMENTS:

1. FOR … TO…NEXT: Count Controlled loop
2. REPEAT … UNTIL : Post Condition loop
3. WHILE…DO…ENDWHILE: Pre-Condition Loop

Problem: Input daily wages and number of day worked and output monthly pay for 100 employees.

Problem: Input marks of a student in a class output result. Passing marks is 40

Problem: Input marks of 30 students in a class output result of each student. Passing marks is 40

Problem: Print name of Allah 10 times
Flowchart of pre-condition and post condition loops
Problem: Print name of Allah 10 times using all types of loops

**Pre-Condition Loop: (WHILE … DO … ENDWHILE)**
When condition to continue the loop is given
→
→
→
Problem: To input and add a series of positive numbers in total. Continue this process for input of positive numbers

| WHILE | |
|---|---|
| | |

**Post-Condition Loop (REPEAT … UNTIL)**
When condition is given at the end of loop
→
→
→
Problem: Input a series of numbers, calculate their total, stop input if total is more than 100

| REPEAT … UNTIL Loop | |
|---|---|
| | |

Differences between

| Pre-Condition | Post Condition |
|---|---|
| | |
| | |
| | |
| | |

**Summer 2017 P22**

**4** An algorithm has been written in pseudo code to input 100 numbers and print out the sum.

A REPEAT … UNTIL loop has been used.

```
Count ←     0
Sum ←       0
REPEAT
        INPUT Number
        Sum ← Sum + Number
        Count ←Count + 1
UNTIL Count > 100
PRINT Sum
```

**(a)** Find the error in the pseudo code and suggest a correction.

Error 1 .....

Correction .....

.....[2]

**(b)** Rewrite the correct algorithm using a more suitable loop structure.

.....

.....

.....

.....

.....

.....

..... [3]

There are three different types of looping structures. Write pseudo code for each of following three problems using different looping structure:

**a)** Input daily temperature for a month of 30 days, calculate and output their total and average.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

…… [6]

***Summer 2018 P22***

**b)** Draw a flowchart for an algorithm to input numbers. Reject any numbers that are negative and count how many numbers are positive. When the number zero is input, the process ends and the count of positive numbers is output.

……

……

……

……

……

……

……

……

……

……

……

……

……

…… [6]

***Extracted from March 2018 P22 (India)***

**c)** Input the weight in kilograms of a passenger stepping into a lift. The lift can take a maximum weight of 640 kilograms. Stop input when total weight is greater than or equal to maximum allowed weight and display message "Ready to Go"

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

**Q 9.6)** Write an algorithm, using pseudo code and a FOR … TO … NEXT loop structure, to input 1000 numbers into an array.

.....

.....

.....

.....

.....

.....[2]

***Summer 2015 P22***

**(b)** Rewrite your algorithm using another loop structure.

.....

.....

.....

.....

.....

.....

.....[4]

Q 9.8a) Write a pseudo code algorithm to input weight of 30 items in kilograms to be loaded on a trailer. Calculate the average weight items under or of 25 kg.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

**Q 9.8b)** Explain how you change your pseudo code to reject any item over 25 Kg.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

## Control Constructs

**Q 1)** Write down different statements for following tasks

| Input | Output | Selection | Iteration |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

**Q 2)** Show what type of programming construct each statement represents.

Complete the table by putting a tick (✓) in the appropriate column for each item.

| Item | Statement | Selection | Iteration | Assignment |
|---|---|---|---|---|
| **1** | MyScore = 65 | | | |
| **2** | FOR IndexVal = 0 TO 99 | | | |
| **3** | MyArray[3] = MID(MyString,3,2) | | | |
| **4** | IF MyScore>= 70 THEN | | | |
| **5** | ENDWHILE | | | |
| **6** | ELSE Message = "Error" | | | |

**Q 3)** Show what type of programming construct each statement represents.

Complete the table by putting a tick (✓) in the appropriate column for each item.

| Item | Statement | Selection | Iteration | Assignment |
|---|---|---|---|---|
| **1** | WHILE DegF> 37.5 | | | |
| **2** | MyName = "Gordon" | | | |
| **3** | DegF = INT(DegF) | | | |
| **4** | ENDIF | | | |
| **5** | CASE OF MyFavourite | | | |
| **6** | UNTIL x = 5 | | | |

**Summer 2015 P22**

**4** Four programming concepts and four examples of programming code are shown below. Draw a line to link each programming concept to the correct example of programming code. [4]

| Programming concept | Example of programming code |
|---|---|
| Counting | `Sum = Sum + Value[n]` |
| Repetition | `IF Value = 10 THEN PRINT 'X'` |
| Selection | `FOR Counter = 1 TO 10` |
| Totalling | `Amount = Amount + 1` |
| | `Sum = Num1 + Num2` |

*Examiner's comments on Question 4*

*Nearly all candidates could link the programming concept of selection with the correct example of programming code. Many candidates correctly linked at least three out of the four programming concepts. Due to an issue with this question, a discussion took place between the Principal Examiner and Assessment specialists to consider the impact on candidates in the light of answers seen. No candidates were disadvantaged and the full range of marks was seen.*

**Summer 2015 P21& 23**

**6** Identify **three** different loop structures that you can use when writing pseudo code.

1.....

2 .....

3..... [3]

*Examiner Report Question 6*

*Most candidates could identify at least one loop structure. A common wrong answer was to incorrectly identify IF as part of a loop structure.*

**Summer 2015 P22**

**5 (a)** Write an algorithm, using pseudo code and a FOR … TO … NEXT loop structure, to input 1000 numbers into an array.

.....

.....

.....

.....

.....

.....[2]

**(b)** Rewrite your algorithm using another loop structure.

.....

.....

.....

.....

.....

.....[4]

*Examiner's comments on Question 5*

*(a) Most candidates attempted the loop structure, better candidates also showed the skill of being able to use the loop counter as the array index. Some candidates misread the question and incorrectly provided program code rather than pseudo code.*

*(b) Better candidates correctly used REPEAT … UNTIL or WHILE … DO … ENDWHILE structures.*

*The most challenging aspect was the correct management of the loop counter.*

**Winter 2015 P21 & 22**

**5** Identify **two** different conditional statements that you can use when writing pseudo code.

1 .....

2 .....[2]

*Examiners' Comments Question 5*

*Many candidates could identify IF as a conditional statement. Candidates with stronger responses throughout also identified CASE.*

**Summer 2016 P21 &P23**

**5** REPEAT ... UNTIL is one type of loop structure.

Identify and describe **two** other types of loop structure that you could use when writing pseudo code.

Loop structure 1: .....

Description: .....

.....

Loop structure 2: .....

Description: .....

.....[4]

## Summer 2016 P22

4 Four statement types and four examples are shown below.
Draw a line to connect each statement type to the correct example.

| Statement type | Example |
|---|---|
| Assignment | `FOR X ← 1 TO 10` |
| Iteration | `READ X` |
| Input | `PRINT X` |
| Output | `X ← Y + Z` |

[3]

## Winter 2016 P21-23

**5** REPEAT ... UNTIL and WHILE ... DO ... ENDWHILE are two different loop structures you can use when writing pseudo code.

Explain, using examples, why you would choose to use each type of loop.

Example 1 .....

.....

.....

.....

Reason for choice .....

.....

.....

Example 2 .....

.....

.....

.....

Reason for choice .....

.....

.....[6]

**Winter 2016 P22**

**4** IF ... THEN ... ELSE ... ENDIF and CASE ... OF ... OTHERWISE ... ENDCASE

are two different conditional statements that you can use when writing pseudo code.

Explain, using examples, why you would choose to use each conditional statement.

Example 1 .....

.....

.....

.....

.....

.....

Reason for choice .....

.....

.....

Example 2 .....

.....

.....

.....

.....

.....

Reason for choice .....

.....

..... [6]

**March 2017 P21 (India)**

**5 (a)** Rewrite the following pseudo code algorithm using a WHILE … DO … ENDWHILE loop.

```
INPUT Num
FOR Counter ← 1 TO 12
        Num ← Num * Counter
        A[Counter] ← Num
NEXT
```

[4]

**(b)** Explain the differences between a WHILE … DO … ENDWHILE and a REPEAT … UNTIL loop .

.....

.....

.....

.....

.....

.....

.....

.....[4]

**Summer 2017 P22**

**4** An algorithm has been written in pseudo code to input 100 numbers and print out the sum. A REPEAT … UNTIL loop has been used.

```
Count ←     0
Sum ←               0
REPEAT
        INPUT Number
        Sum ← Sum + Number
        Count ←Count + 1
UNTIL Count > 100
PRINT Sum
```

**(a)** Find the error in the pseudo code and suggest a correction.

Error.....

Correction .....

.....[2]

**(b)** Rewrite the correct algorithm using a more suitable loop structure.

.....

.....

.....

.....

.....

.....

.....[3]

**Summer 2017 P21**

**5 (a)** Describe the purpose of each statement in this algorithm.

```
FORI← 1 to 300
        INPUT Name[I]
NEXT I
```

.....

.....

.....

.....

.....[2]

**(b)** Identify, using pseudo code, another loop structure that the algorithm in part (a) could have used.

.....

.....[1]

**(c)** Write an algorithm, using pseudo code, to input a number between 0 and 100 inclusive. The algorithm should prompt for the input and output an error message if the number is outside this range.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[3]

**Winter 2017 P21**

**4 (a) Four** pseudo code descriptions and **five** pseudo code statements are shown. Draw one line to link each pseudo code description to the correct pseudo code statement. Not all pseudo code statements will be used.[4]

| Pseudo code description | Pseudo code statement |
|---|---|
| A loop that will iterate at least once. | FOR…TO…NEXT |
| A conditional statement to deal with many possible outcomes. | IF…THEN…ELSE…ENDIF |
| A loop that will iterate a set number of times. | WHILE…DO…ENDWHILE |
| | CASE…OF…OTHERWISE…ENDCASE |
| A conditional statement with different outcomes for true and false. | REPEAT…UNTIL |

**Winter 2017 P22**

**4** IF … THEN … ELSE … ENDIF is one type of conditional statement used when writing pseudo code.

Identify and describe **another** type of conditional statement that you could use when writing pseudo code. Give a reason why you would use this type of conditional statement.

Conditional statement .....

.....

Description .....

.....

.....

.....

Reason .....

..... [4]

**March 2018 P22 (India)**

**5** Explain the difference between the programming concepts of **sequence** and **selection**. Include an example of a programming statement for each concept in your explanation.

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

***Comments on Question 5***

*Candidates found the explanation of the difference between the programming concepts **sequence** and **selection** challenging, with few candidates identifying that programming statements in a sequence were executed one after another whilst selection meant that the path through the program depends on the result of a question. Candidates were more successful in providing suitable examples of programming statements.*

*Common errors included confusing sequence or selection with iteration.*

**Summer 2018 P21**

**5** Explain the difference between the programming concepts of **counting** and **totalling**. Include an example of a programming statement for each concept in your explanation.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

**Winter 2018 P22**

**4** A programmer wants to test that the readings from 2000 electricity meters are greater than 400units and less than 900 units. The programmer uses selection and repetition statements as part of the program. Explain, using programming statements, how selection and repetition could be used in this program.

Selection .....

.....

.....

.....

.....

Repetition.....

.....

.....

.....

.....

.....

.....

.....[4]

**Winter 2018 P23**

**3** Give an example of a pseudocode statement or statements to perform each of the following functions.

A condition controlled loop .....

.....

.....

A conditional statement .....

.....

.....

Totalling .....

.....

.....[3]

**Winter 2018 P22**

**4** A programmer wants to test that the readings from 2000 electricity meters are greater than 400units and less than 900 units. The programmer uses selection and repetition statements as part of the program. Explain, using programming statements, how selection and repetition could be used in this program.

Selection .....

.....

.....

.....

Repetition .....

.....

.....

..... [4]

**March 2019 P22**

**4** For each of the **four** groups of statements in the table, place a tick in the correct column to show whether it is an example of **Selection** or **Repetition**. [4]

| Statements | Selection | Repetition |
|---|---|---|
| FOR X ← 1 TO 10<br>  SUM ← SUM + 1<br>NEXT X | | |
| WHILE X > 10 DO<br>  SUM ← SUM + 1<br>  X ← X - 1<br>ENDWHILE | | |
| IF X > 10<br>  THEN<br>  SUM ← SUM + 1<br>  X ← X - 1<br>ENDIF | | |
| REPEAT<br>  SUM ← SUM + 1<br>  X ← X - 1<br>UNTIL X > 10 | | |

**Summer2019 P22**

**4** For each of the **four** groups of statements in the table, place a tick in the correct column to show whether it is an example of **Selection** or **Repetition**. [4]

| Statements | Selection | Repetition |
|---|---|---|
| FOR A ←1 TO 100<br>B ←B + 1<br>NEXT A | | |
| CASE A OF<br>100: B ←A<br>200: C ←A<br>ENDCASE | | |
| IF A > 100<br>THEN<br>B ←A<br>ENDIF | | |
| REPEAT<br>A← B * 10<br>UNTIL A > 100 | | |



**Summer2019 P21**

**3 (a)** Give an example of a conditional statement using pseudocode.

.....

.....

.....

.....

..... [2]

**(b)** Describe the purpose of a conditional statement.

.....

.....

.....

..... [2]

**Summer 2016 P22**

**2** Read this section of program code that inputs 10 positive numbers and then outputs the total.

```
Total = 0
Counter = 0
REPEAT
    INPUT Num
    Total = Total + Num
    PRINT Total
    Counter = Counter + 1
UNTIL Counter = 10
```

This code works, but it is inefficient.

**(i)** Suggest **three** improvements that could be made.

1 .....

.....

2 .....

.....

3 .....

.....[3]

**(ii)** Rewrite the program code with your improvements.

.....

.....

.....

.....

.....

.....

.....

.....

.....[3]

# Algorithm pseudo code

**Q 12.1) Summer 2006 (Extract)**
A formula for calculating the body mass index (BMI) is:

$$\text{BMI} = \frac{\text{weight in kilograms}}{(\text{height in metres}) \times (\text{height in metres})}$$

Using pseudo code or otherwise, write an algorithm that will input weight (kg) and height (m) of students, calculate their body mass index (BMI) and output their BMI.
Test data: 80, 2, 100, 1.9, 60, 2, 70, 1.8

**First draw trace table write down column headings**
**Calculate BMI using trace table:**

| Weight | Height | BMI |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Setup in pseudo code using declaration of variable
**SECTION SHOWS YOU HOW THIS WOULD WORK'**

**Now Input Weight and height**

| Weight | Height | BMI |
|---|---|---|
| 80 | 2 | |
| 100 | 1.9 | |
| 60 | 2 | |
| 70 | 1.8 | |

Input in pseudo code using test data

**Now calculate the BMI using given formula**

| Weight | Height | BMI |
|---|---|---|
| 80 | 2 | 20 |
| 100 | 1.9 | 28 |
| 60 | 2 | 15 |
| 70 | 1.8 | 22 |

Process in pseudo code using given formula



**Now write down the above steps in pseudo code:**

```
DECLARE Weight, Height, BMI: Real
INPUT Weight, Height
BMI ← Weight/(Height*Height)
OUTPUT BMI
```

**Q12.2)Winter 2007 (Extract)**
Fuel economy for a car is found using the formula:

$$\text{Fuel Economy} = \frac{\text{Distance Travelled (km)}}{\text{Fuel Used (litres)}}$$

Using pseudo code or otherwise, write an algorithm that will input Distance Travelled (km) and Fuel Used (litres) of cars, calculate their fuel economy and output their fuel economy.
Test data: 80, 10, 100, 5, 60, 2, 70, 5

**First draw trace table write down column headings**

| Distance | Fuel | Fuel Economy |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Setup in pseudo code using declaration of variable

**Now Input Distance and Fuel**

| Distance | Fuel | Fuel Economy |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Input in pseudo code using test data

**Now calculate the Fuel Economy using given formula**

| Distance | Fuel | Fuel Economy |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Process in pseudo code using given formula

**Now write down the above steps in pseudo code:**

**Q12.3)** Write an algorithm, using pseudo code or flowchart only, which:

- inputs real numbers
- convert them into integer (whole) numbers

(You may use INT(X) in your answer e.g. Y = INT(3.8) gives the value Y = 3)

Test data: 80.9, 10.1, 100.8, 5.6

**First draw trace table write down column headings**

| Number X | Integer Y | Output |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Setup in pseudo code using declaration of variable

**Now Input Number**

| Number X | Integer Y | Output |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

Input in pseudo code using test data

**Now convert the real number into whole number using INT()**

| Number X | Integer Y | Output |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

**Now write down the above steps in pseudo code:**

**Q12.4)** Write an algorithm, using pseudo code or a program flowchart only, which:

• inputs the population and land area,

• calculates the population density (i.e. population/land area),

**Q12.7)**This code is supposed to find out if a positive integer entered by a user is exactly divisible by the number 3.
**Note**: line numbers have been included and are not part of the code.

```
1    INPUT n
2    WHILE n ≥ 0
3        n ← n – 3
4    ENDWHILE
5    IF n = 0 THEN
6        OUTPUT 'is divisible by 3'
7    ELSE
8        OUTPUT 'is not divisible by 3'
9    ENDIF
```

The programmer realizes there is an error because a user input of 6 incorrectly outputs 'is not divisible by 3'.
**(a)** In **Table** place a tick next to the type of error that the programmer has found. **[1]**

| Type of error | Tick |
|---|---|
| Logical | |
| Runtime | |
| Syntax | |

**(b)** State the line number of the code containing the mistake that causes this error to occur.

.....**[1]**

(c)What change needs to be made to the line of code you have identified in your answer to(b) so that the program will work correctly?

..... **[1]**

(d)What type of error could occur if the user enters the value eight?

.....**[1]**

**12.7**

*a Logical*

*b2*

*c Any correct answer, examples include:*
*If the answer given for 9 (b) is 4 then*
*WHILE n > 0*
*WHILE n ≥ 1*
*WHILE n ≥ 3*
*If the answer given for 9 (a) (ii) is 7 then*
*IF n = -3 THEN*
*d Runtime error // Type error*

**Q12.8)** The following pseudo code calculates the second hand price of different models of car.

The condition is an integer with a value between 1 and 4 where 1 is excellent and 4 is very bad.

```
INPUT Model, Condition, Age
cost ← 0
IF model = 'Daley' THEN
        cost ← 6000
ELSE IF model = 'Minty' THEN
        cost ← 4000
ELSE
        cost ← 2000
ENDIF

CASE condition OF
1: cost ← cost – 100
2: cost ← cost – 300
3: cost ← cost – 500
4: cost ← cost – 1000
ENDCASE
cost ← cost / age
PRINT cost
```

a) Tick the most appropriate data type of the variable cost. [1]

| **Data Type** | **Tick one box** |
|---|---|
| Boolean | |
| Character | |
| Real | |
| String | |

**b)** Complete the trace table below showing the changes in the variable cost when the following values are input: **[4]**

"Tidy", 4, 2

| Cost |
|---|
| |
| |
| |
| |

**12.8**

*a) Real*

*b) 1 mark for every correct row that appears in the correct sequence:*

| ***cost*** |
|---|
| *0* |
| *2000* |
| *1000* |
| *500* |

**Q12.9)** Write an algorithm, using pseudo code or flowchart only, which:

• inputs 1000 numbers

• outputs how many of these numbers were whole numbers (integers)

(You may use INT(x) in your answer, e.g. y = INT(3.8) gives the value y = 3)

..... [4]

**(You may use INT(x) in your answer, e.g. y = INT(3.8) gives the value y = 3)**

Y=INT(X)
INT function
removes
fractional part

| INPUT X | Y=INT(X) | Is X=Y? | CountINT |
|---|---|---|---|
| | | | 0 |
| 3.8 | 3 | No | |
| 4 | 4 | Yes | 1 |
| 5 | 5 | Yes | 2 |
| 9.1 | 9 | No | |
| 7 | 7 | Yes | 3 |

Initial value
CountINT← 0

CountINT←CountINT+1
Increment if X is an integer

For integer
numbers X and y
will be equal

**12.9**

```
DECLARE Count, CountINT : Integer
DECLARE X, Y: Real
CountINT← 0
FOR Count ← 1 TO 1000
        PRINT " Enter a number "
        INPUT X
        Y ← INT(X)
        IF X = Y THEN CountINT←CountINT + 1
NEXT Count
PRINT " Number of integers = " , CountINT
```

**Q12.10)**

**Q12.18)** A programmer uses an Integrated Development Environment (IDE) for all program development. Describe what is meant by an IDE.

.....

.....

..... [2]

**12.18**

*IDE is a (Single) software program*
*Features for:*
*program editor/writing/editing*
*translation // interpreter/compiler*
*testing program code // observe outputs  2 points to score*

**Q 12.23)** An algorithm to reset the contents of the array Coins after each sale is shown below. The re are 10 different coins. This algorithm contains a logic error.

```
i = 1
REPEAT
        Coins(i) = 0
        i = i + 1
UNTIL i = 10
```

**(i)** State what is meant by a logic error.

.....

..... **[1]**

**(ii)** Explain why the algorithm above contains a logic error.

.....

.....

.....

..... **[2]**

*(i) •The program is written to do something other than what the programmer intended*

*(ii) •It will only reset the first 9 elements / will not reset the 10th element*
*•After setting Coins(9) = 0, i will become10...*
*•... and the loop will stop*
*•It should be UNTIL i > 10 / or other working correction*

## Quick Revision Questions of flowchart and pseudo code

**Q9.4)** Identify **three** different loop structures that you can use when writing pseudo code.

1.....

2 .....

3..... [3]

*Most candidates could identify at least one loop structure. A common wrong answer was to incorrectly identify IF as part of a loop structure.*

**Q .12** Describe the purpose of each statement in this algorithm.

```
Count ← 0
WHILE Count<10 DO
        PRINT Count
        Count ←Count + 1
ENDWHILE
```

.....
.....
.....
.....
.....[2]

**9.13** Describe the purpose of each statement in this algorithm.

```
Total ← 0
INPUT Number
REAPEAT
        Total ←Total + Number
        INPUT Number
UNTIL Number < 0
PRINT Total
```

.....
.....
.....
.....
.....
.....
.....[2]

**Q 9.14a)** Draw a flowchart to input 20 numbers and find the average of positive numbers

**Q 9.14b)** Explain how do you change your flowchart to work for 30 numbers that are between 0 an d 100.

.....

.....

.....

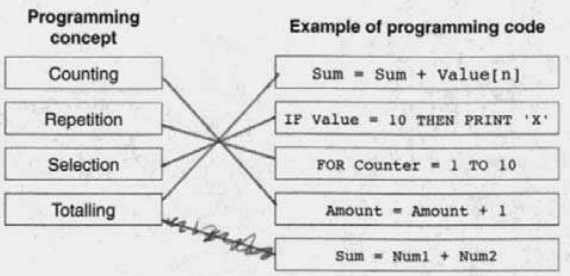
.....[3]

## Candidate Example response

### Example candidate response – high

**4** Four programming concepts and four examples of programming code are shown below.

Draw a line to link each programming concept to the correct example of programming code.




[4]

### Examiner comment – high

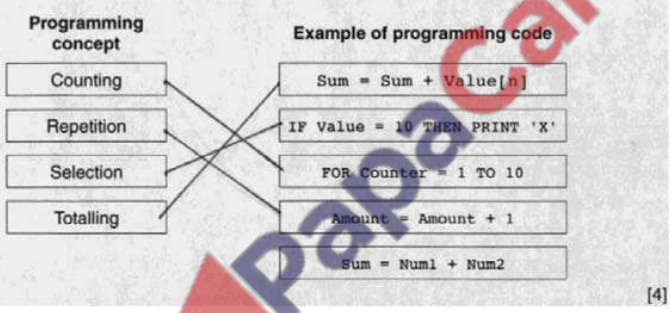
Most of the high-awarding candidates gained full marks.

**Total mark awarded = 4 out of 4**

### Example candidate response – middle

**4** Four programming concepts and four examples of programming code are shown below.

Draw a line to link each programming concept to the correct example of programming code.




[4]

### Examiner comment – middle

Most of the middle-awarding candidates could identify 'selection' and one other programming concept.

**Total mark awarded = 2 out of 4**

## Example candidate response – low

**4** Four programming concepts and four examples of programming code are shown below.

Draw a line to link each programming concept to the correct example of programming code.

| Programming concept | Example of programming code |
|---|---|
| Counting | `Sum = Sum + Value[n]` |
| Repetition | `IF Value = 10 THEN PRINT 'X'` |
| Selection | `FOR Counter = 1 TO 10` |
| Totalling | `Amount = Amount + 1` |
| | `Sum = Num1 + Num2` |

[4]

## Examiner comment – low

Most of the low-awarding candidates could only identify the programming concept of 'selection'.

**Total mark awarded = 1 out of 4**

## Example candidate response – high

5 (a) Write an algorithm, using pseudocode and a FOR … TO … NEXT loop structure, to input 1000 numbers into an array.

```
number[1:1000]
for counter = 1 to 1000
    input num
    number[counter] = num
next counter
```

## Examiner comment – high

A FOR … TO … NEXT loop with correct use of the loop counter for the array index, full marks.

**Total mark awarded = 2 out of 2**

## Example candidate response – middle

5 (a) Write an algorithm, using pseudocode and a FOR … TO … NEXT loop structure, to inpu 1000 numbers into an array.

```
num=0
For count = 1 to 1000
    Input num
Next
Numbers [1:1000] as integer
Numbers [x] <- N
```

[2

## Examiner comment – middle

A FOR … TO … NEXT loop, there is no attempt to use the loop counter with the array.

**Total mark awarded = 1 out of 2**

**5** **(a)** Write an algorithm, using pseudocode and a FOR … TO … NEXT loop structure, to input 1000 numbers into an array.

INPUT = 1000.
FOR,
1000 > input
TO,
9999 > input.
Next, PRINT Output [2]

## Examiner comment – low

An attempt at a FOR … TO … NEXT loop, there is no loop counter and no use of an array.

**Total mark awarded = 0 out of 2**

## Example candidate response – high

**(b)** Rewrite your algorithm using another loop structure.

```
Number [1:1000] , count ← 0
Repeat
Input num
~~Not~~ count ← count + 1
Number [count] ← num
Until count = 1000
```

## Examiner comment – high

A REPEAT … UNTIL loop, with correct initialisation, updating and testing of the loop counter, full marks. The candidate has used the correct ← symbol as required by the new syllabus. Candidates using = instead of ← were not penalised.

**Total mark awarded = 4 out of 4**

## Example candidate response – middle

**5 (a)** Write an algorithm, using pseudocode and a FOR … TO … NEXT loop structure, to input 1000 numbers into an array.

```
num=0
For count = 1 to 1000
  Input num
Next
Numbers [1:1000] as integer
Numbers [x] ← N
```

[2]

## Examiner comment – middle

A WHILE … DO … ENDWHILE loop, with some errors. The loop counter has not been initialised, the WHILE statement is missing a variable. The updating of the loop counter is correct and there is an ENDWHILE statement, for two marks.

**Total mark awarded = 2 out of 4**

## Example candidate response – low

**(b)** Rewrite your algorithm using another loop structure.

```
Numbers = [1 to 1000]
Input number
c = c + 1
Next
If the numbers = 1000 then
print Yes
display number
```

[4]

## Examiner comment – low

It is unclear what type of loop is being used here. There is one mark for updating the loop counter.

**Total mark awarded = 1 out of 4**

**Past paper questions on basic concepts of algorithm**

**Specimen paper 2016 P2**

2 Jatinder uses Internet banking.

This pseudo code checks her PIN.

```
c ← 0
INPUT PIN
x ← PIN
REPEAT
    x ← x/10
    c ← c + 1
UNTIL x < 1
IF c <> 5
    THEN
        PRINT "error in PIN entered"
    ELSE
        PRINT "PIN OK"
ENDIF
```

(a) What value of c and what message would be output if the following PINs were entered?

5 1 0 2 0 Value of c: .....

Message:.....

5 1 2 0 Value of c:.....

Message: ..... [2]

(b) What type of validation check is being carried out here?

..... [1]

**Specimen paper 2016 P2**

6 (a) Write an algorithm, using pseudo code or flowchart only, which:

• inputs three numbers

• outputs the largest of the three numbers

.....

.....

.....

.....

.....

..... [3]

(b) Write an algorithm, using pseudo code or flowchart only, which:

• inputs 1000 numbers

• outputs how many of these numbers were whole numbers (integers)

(You may use INT(x) in your answer, e.g. y = INT(3.8) gives the value y = 3)

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) Describe, with examples, two sets of test data you would use to test your algorithm.

1: .....

.....

2: .....

..... [2]

## Summer 2016 P22

6 Identify two different selection statements that you can use when writing pseudo code.

1 ....................................................................................................................................

....................................................................................................................................

2 ....................................................................................................................................

....................................................................................................................................[2]

## Winter 2017 P22

**2** Write an algorithm using **either** pseudo code **or** a flowchart, to:

- input a positive integer
- use this value to set up how many other numbers are to be input
- input these numbers
- calculate and output the total and the average of these numbers.

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................................................

....................................................................................................[6]

## Winter 2017 P22

**3** The following diagram shows **four** data structures and **four** descriptions. [3]

Draw a line to connect each data structure to the correct description.

| Data structure | Description |
|---|---|
| Constant | A collection of related data |
| Array | A value that can change whilst a program is running |
| Table | A value that never changes whilst a program is running |
| Variable | A series of elements of the same data type |

**Question 10** **Count**

A small airport handles 400 flights per day from three airlines:

FASTAIR (code FA)
SWIFTJET (code SJ)
KNIGHTAIR (code KA)

Each flight is identified by the airline code and 3 digits. For example FA 156.

Write an algorithm, using pseudocode or otherwise, which monitors the 400 flights into and out of the airport each day. The following inputs, processing and outputs are all part of the monitoring process:

- input flight identification
- calculate number of flights per day for each of the three airlines
- output the percentage of the total flights per day by each airline
- any validation checks must be included

```
DECLARE CountFA, CountSJ, CountKA: Integer
DECLARE AirlineCode, Count: Integer
CoutFA <-- 0
CoutSJ <-- 0
CoutKA <-- 0
FOR Count<-- 1 TO 400
        INPUT AirLineCode
        WHILE AirLineCode<>"FA" AND AirLineCode<>"SJ" AND  AirLineCode<>"KA" DO
                PRINT "Enter a valid air line code"
                INPUT AirLineCode
        ENDWHILE
        INPUT FlightCode
        WHILE FlightCode<100 OR FlightCode>999 DO
                PRINT" Error! Enter a valid flight code"
        ENDWHILE
        IF AirLineCode="FA" THEN CountFA <-- CountFA + 1
        IF AirLineCode="SJ" THEN CountSJ <-- CountSJ + 1
        IF AirLineCode="KA" THEN CountFA <-- Count KA + 1
NEXT  Count
FAPercent <-- CountFA/400*100
SJPercent <-- CountSJ/400*100
KAPercent <-- CountKA/400*100
PRINT " Number of FastAir ", CountFA
PRINT " Number of SWIFJET ", CountSJ
PRINT " Number of KNIGHTAIR", CountKA
```

**Summer 2018 P21**

**2 (a)** Write an algorithm to input 1000 numbers. Count how many numbers are positive and how many numbers are zero. Then output the results. Use **either** pseudo code **or** a flowchart.

..... [6]

**(b)** Give one change you could make to your algorithm to ensure initial testing is more manageable.

..... [1]

**Summer 2018 P22**

**2 (a)** Draw a flowchart for an algorithm to input numbers. Reject any numbers that are negative and count how many numbers are positive. When the number zero is input, the process ends and the count of positive numbers is output.



**(b)** Explain the changes you will make to your algorithm to also count the negative numbers.

.....

.....

.....

.....[2]

**Q 12.79 Winter 2018 P23**

**5** The algorithm allows a number to be entered. It then calculates and outputs the next number in the mathematical series.

```
Fib ←1
Prev2 ←0
Prev1 ←1

INPUT Number

IF Number = 0
        THEN Fib = 0
ENDIF
WHILE Number > 2
        Fib ←Prev2 + Prev1
        Prev2 ←Prev1
        Prev1 ←Fib
        Number ←Number-1
ENDWHILE
OUTPUT Fib
```

**(a)** Complete the trace table for the input data: **7** [4]

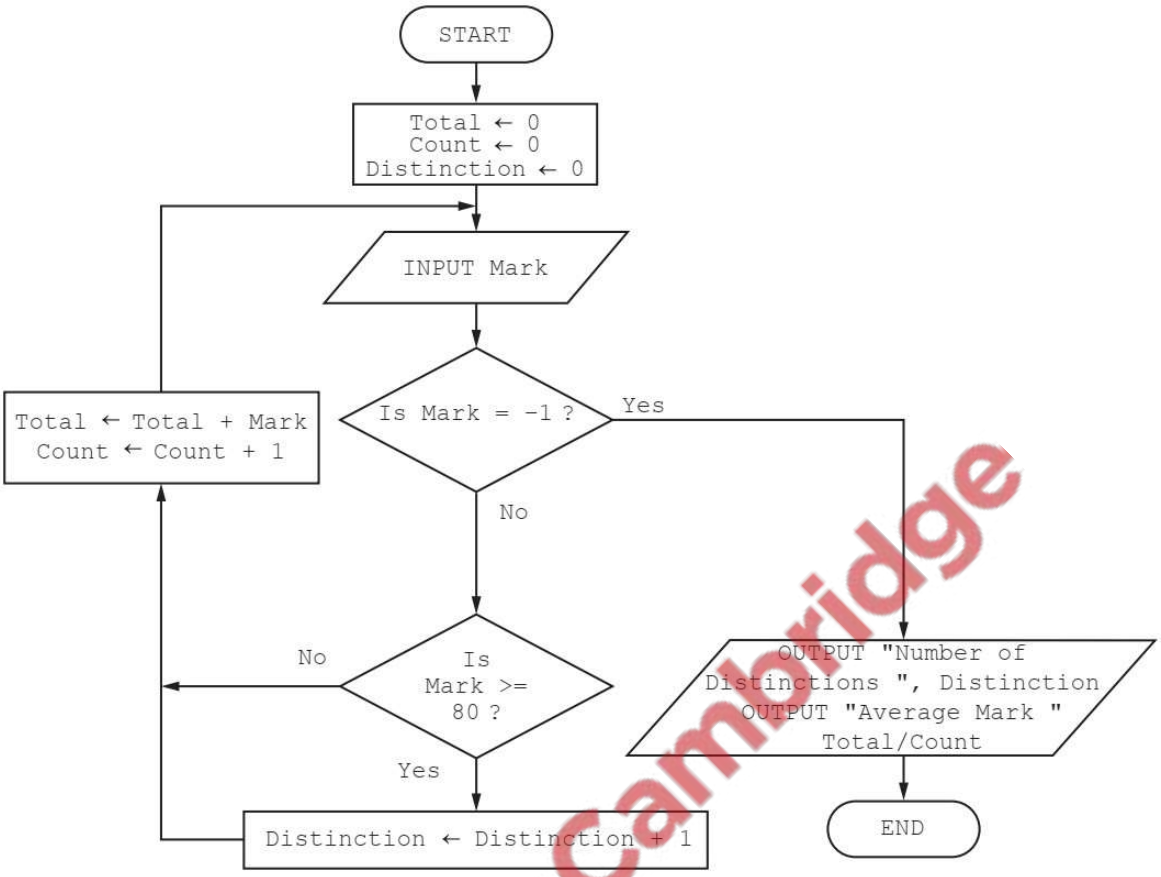
| Fib | Prev2 | Prev1 | Number | OUTPUT |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**(b)** Complete the trace table for the input data: **2** [2]

| Fib | Prev2 | Prev1 | Number | OUTPUT |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Q 12.83 Summer2019 P22**

**3** This flowchart inputs the marks gained in an examination. An input of –1 ends the routine.




Complete the trace table for the mark input data: 50, 70, 65, 30, 95, 50, 55, 85, 65, 35, –1, 45[4]

| Total | Count | Distinction | Mark | **OUTPUT** |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Linear Search**

5 Customer names are stored in the array Customer.

An algorithm is to be designed to perform a serial search of the array for a requested customer name.

The algorithm will use the variables shown in the table.

(a) Study the table and the algorithm and fill in the gaps.

| Identifier | Data Type | Description |
|---|---|---|
| Customer | ARRAY[100] OF STRING | Array of customer names |
| Index | INTEGER | Used to index the array elements |
| IsFound | | |
| SearchName | STRING | The requested customer name |

```
//Serial search algorithm
INPUT ……..…

IsFound ← FALSE
Index ← 1
REPEAT

        IF …….……  =SearchName

        THEN

                IsFound ← TRUE

                OUTPUT "Found at position " Index

        ELSE

                …….……..

        ENDIF

UNTIL (IsFound = TRUE) OR ……..

IF …….…….. THEN

        OUTPUT "Customer name was NOT FOUND"

ENDIF
```

[7]

(b) How many comparisons on average will be needed to find a requested customer from the Customer array?

……[1]

3 (a) Customer names are stored in the array Customer.

An algorithm is to be designed to perform a serial search of the array for a requested customer na me.

The algorithm will use the variables shown in the table.

Study the table and the algorithm and fill in the gaps.

| Identifier | Data Type | Description |
|---|---|---|
| Customer | ARRAY[2000] OF STRING | The customer names |
| Index | INTEGER | Index position in the customer array |
| IsFound | | |
| SearchName | STRING | The requested customer name |

```
//Serial search algorithm
INPUT ……
IsFound ← FALSE
Index ← 1
REPEAT
        IF Customer [ ……] = SearchName        THEN
                IsFound ← TRUE
                OUTPUT "FOUND – at position " Index " in the array"
        ELSE
                Index ← ……
        ENDIF
UNTIL (IsFound = TRUE) OR
IF ……..…….  THEN
        OUTPUT "Customer name was NOT FOUND"
ENDIF
```

[7]

(b) Comment on the efficiency of the serial search algorithm in part (a) for retrieving a data item from an array with 2000 items.

.....

..... [2]