

Unit 3. Elementary Principles of Programming

Introduction, Definition of terms

Introduction

Programming is done by use of special characters, signs and symbols found in a particular programming language. Programming language can be translated into machine readable form by the computer

- a) **Programming** – The process of creating a list of instructions that a computer will use to execute a certain task.
- b) **Programming language** – A set of special symbols and characters that can be translated into machine readable form used to create programs.
- c) **Source program** – A code of program that has not been translated into machine readable form.
- d) **Object code** – A program code that has been translated into machine readable form.
- e) **Language translator** – Language processors or utility programs that convert source code to object code.
- f) **Assembler** – A language translator that converts assembly language into machine code.
- g) **Interpreter** – A language translator that converts and executes high level source code line-by-line.
- h) **Compiler** – A language translator converts the entire high level source code to object code at once.

Interpreter	Compiler
Translates one statement at a time	Translates entire source code at once
Takes less memory	Requires more memory
Translates each time it is run	Object code can be stored and run as required
Slower than compiling	Faster than translating

Levels of programming languages

There are two major levels. These are:

- i) Low-level languages
- ii) High-level languages

i) Low-level languages

There are two types of low-level languages. These are:

a) Machine (1st generation) language

Instructions are written in binary form (1's and 0's) containing both opcode (instructions) and operands (data items).

Advantages of low-level languages

- No need for translation.
- Execution of instructions is fast.
- Economises on memory.
- Can be used to develop secret codes.

Disadvantages of low-level languages

- Not portable – machine dependent.
- Difficult to learn or use thus requires highly trained programmers to develop or maintain.
- Difficult to debug.

b) Assembly (2nd generation) language Uses English acronyms called mnemonics such as ADD, MOVE, SUB and LOAD.

Advantages of assembly language

- Easy to understand than machine code.
- Fast to debug.
- The program codes are shorter than machine code.

Disadvantages of assembly language

- Machine dependent.
- Needs a lot of training to understand.
- Needs translation therefore slow to execute.

ii) High-level languages

- They use a language very close to natural or ordinary human language.
- They are therefore very user friendly.
- Are machine independent and can be classified as follows:
 - a) Third generation languages.
 - b) Fourth generation languages.
 - c) Fifth generation languages.
 - d) Object oriented languages (OOPs).
 - e) Web scripting languages.

a) Third generation languages

- Also called structured or procedural languages.
- They allow a program to be broken into parts called modules, each of which performs a specific task.
- Program control structures are used in program development to dictate the logic of a program.

Examples include:

- i) **Pascal** – An academic language for teaching structured programming.
- ii) **FORTTRAN** (FORmula TRANslator) – For coding mathematical and scientific applications.
- iii) **COBOL** – (Common Business Oriented Language) - For developing business or commercial applications.
- iv) **BASIC** – (Beginners All-purpose Symbolic Instructional Code) - Used for developing business and educational applications. Also used to teach structured programming.
- v) **C** – Used to develop system software that permits direct control over the hardware, for example operating systems.
- vi) **Ada** – For developing real - time systems
- vii) **LOGO** – For educational purpose to teach how to program.

b) Fourth generation languages (4GLs)

- They present a programmer with more programming tools such as command buttons and forms.
- Allow programmers to select graphical objects called controls to create designs on a base form.
- They also free programmer from writing codes since they have application generators that generates the codes.
- Examples: Delphi Pascal, Visual Basic and Visual COBOL.

c) Fifth generation languages

- Used to design artificial intelligence applications.
- Enables a computer to mimic human intelligence.
- Programmer worries only about what problem needs to be solved and what conditions need to be met without necessarily developing algorithms to solve them.

Examples: Mercury, PROLOG, OCCAM and LISP.

d) Object oriented programming languages (OOPs)

Object oriented programming languages (OOPs) Have the following characteristics:

- **State** - Each object has specific data values unique to itself.
- Allow a program to be perceived as having various objects interacting to make a whole program.
- **Behaviour** - Each object has a set of things it can accomplish.
- **Encapsulation** - The process of combining data and the functions that operate on the data within an object.
- **Polymorphism** – Use of one interface to things of different types or using a single representation to show things of different types.
- **Inheritance** - New objects take on the characteristics of objects already there.
- **Examples:** Simula, C++, SmallTalk and Java.
- Used in developing Graphical User Interface (GUI) systems and application software.

e) Web scripting languages

- Used to create or add dynamic functionalities on web pages.
- HTML has limitations and is not used alone when developing functional websites.
- Special blocks of code known as scripts may be inserted using languages like JavaScript, VBScript and Hypertext Preprocessor.
- The most commonly used languages are Javascript, HyperText Markup Language (HTML), Hypertext Preprocessor (PHP) and Python.

HTML uses tags which are special words enclosed in < and > symbols which the browser can interpret as commands

Common characteristics of scripting languages include:

- i) Allow complex tasks to be done in a few steps.
- ii) Have integrated compile and run functionality.
- iii) Simple syntax.
- iv) Allow minimum programming language knowledge.
- v) Allow use of a variety of text editors in creation and editing of web pages.

Advantages of high level languages

- i) They are portable from one computer to another.
- ii) They are user friendly and easy to use and learn.
- iii) They are more flexible hence enhance the creativity of the programmer and increase the productivity in the workplace.
- iv) They are easy to correct errors (debug).

Disadvantages of high level languages

- i) They use large memory of the computer.
- ii) Must be interpreted or compiled to machine readable form before the computer can execute them.
- iii) Instructions are many and complex leading to slower program processing

Program development

Program development task is broken into the following stages:

1. Problem recognition

- A problem that requires programmer's attention is identified to exist
- The programmer seeks to understand or interpret it.

2. Problem definition

- Specify objectives and users.
- Specify desired output – Output devices and formats.
- Specify desired input – Input devices and formats.
- Specify desired processing – Type of computers, operating system and type of processing.
- Study the feasibility of implementing the program (economic, legal, operational, technical and schedule).

3. Program design

- Program logic is determined using top down design or modularisation.
- Use of program design tools to represent the logic: pseudo codes and flowcharts.
- Design input screens, reports and storage files.

4. Program coding

- Choose the programming language to use.
- Convert the program design into a working program.
- Follow syntax as the program is coded.

Characteristics of a user friendly program

- i) Well indented.
- ii) Has inline comments.
- iii) Use of variable and constant names that are suggestive of their use.
- iv) Non-ambiguous program statements.
- v) Modularise the program.
- vi) Short and precise program statements.

5. Program testing and debugging

- Involves detection, location and correction of errors.

Two types of errors:

- i) **Syntax errors** – Result from improper use of programming language. Can be detected by translators. A program with syntax errors cannot execute.
- ii) **Logical errors** – Result from improper use of control structures or functions cannot be detected by translators. A program with them will execute but abort prematurely or give wrong results.

Methods of error detection

- i) **Desk checking or dry running** – Picking errors on the program while still on paper before transcription.
- ii) **Using debugging utilities** – Running translators helps detect and locate syntax errors.
- iii) **Running test data** – Used to do trial runs. Normal, extreme and exceptional data should be used.

6. Program implementation and maintenance

- Delivery and installation of the program for use.
- It also involves user training and file conversion.
- There are four ways of converting to the new system.

These are:

- i) **Parallel** – Both current and new programs run simultaneously until the new one proves reliable.
 - ii) **Direct** – The new program completely replaces the current system immediately it is ready.
 - iii) **Phased** – The new program is introduced in stages.
 - iv) **Pilot** – The entire new program is tried on a part of the organisation.
- **Review and maintenance** – A post implementation support to make sure the program is error free, updated and used appropriately accommodating any user changing needs.

7. Program documentation

- It is an on-going stage throughout the development process.
- Involves writing support materials. There are three types of documentations.

These are:

- i) **User oriented** – Gives procedures for use of the program.
- ii) **Operator oriented** – Helps install and maintain the program.
- iii) **Programmer oriented** – Provides technical information to allow modification of the program in future.

Program Control Structures

- They are a block of statements that determine the logical flow of a program.
- There are three types:

1. Sequence

- Statements are executed in the order they are written or coded.
- No branching or looping.

2. Selection

- There are alternatives presented.
- Execution of an option is determined by evaluation of a given condition.
- There are four constructs used to implement selection.

These are: i) If ... then ii) If Then Else iii) Nested If iv) Case

3. Repetition

It is also called iteration or looping.

- Statement(s) are executed several times.
- The number of times repetition occurs is determined by the given condition.
- There are three constructs:
 - i) While...Do
 - ii) Repeat ...Until
 - iii) For...Do

Development of Algorithms

An **algorithm** is a set of finite and logical steps followed in solving a problem. Programming problem solution can be designed using the following tools:

Pseudocode

- Are a set of statements written in English-like language.
- They describe the logic or processing flow of a program.

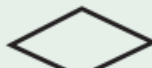
Characteristics of a good pseudocode

- i) Composed of short and clear statements.
- ii) Has well outlined or indented statements.
- iii) Use of non-ambiguous statements.
- iv) Should show start and end of executable statements.
- v) Depict input, processing and output statements using keywords like input or read, output or print, calculate among others.

Flow Charts

- It is a diagrammatic representation of a detailed series of steps needed to solve a programming problem.
- There are two types of flowcharts.
 - i) **System flowchart** – Represents the overall functionality of an information system.
 - ii) **Program flowchart** – Shows the flow of data from input to output stage in a program execution.
- They use standard symbols. The most commonly used symbols are described below.

The Standard Symbols

Symbol	Name	Function
	Terminator	Marks the beginning or the end of a program.
	Input or Output	Shows where data or information is input or output.
	Process	Shows initialisation of values and processing functions or operations.
	Decision	Shows where alternative operations are chosen based on given conditions.
	Onpage connector	Shows movement to or from another part of the program on the same page.
	Off page connector	Shows movement to or from another part of the program in a different page.
	Flow line	Shows the direction of processing logic flow.
	Display	Shows on-screen output.
	Document	Shows paper or hardcopy output.

Characteristics of a good flowchart

- i) Should have only one entry and exit points of the program.
- ii) No lines should cross each other.
- iii) Lines must be straight not curved.
- iv) Symbols should be used for correct purpose.
- v) Logical flow should be clearly indicated using arrows.
- vi) Statements within the symbols should be short and precise.

Developing Pseudocode and Flowchart

1. Using sequence control structure

- Statements are executed in the order they appear on the code.

General pseudocode syntax

Statement 1

Statement 2

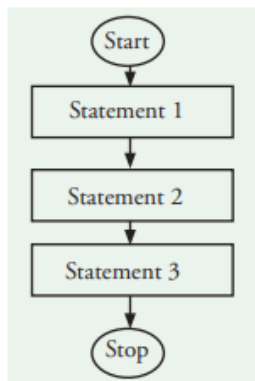
.

.

.

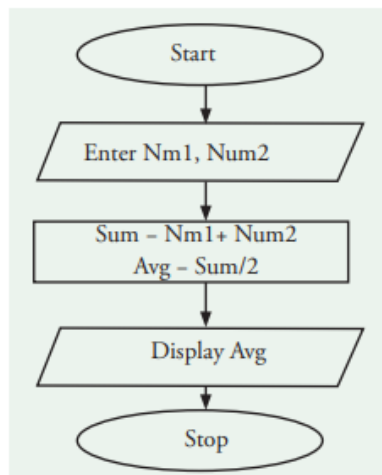
Statement n

Flowchart extract



Example: Draw a program flowchart and write a pseudocode which allows a user to enter marks for two subjects, calculate total and display average score.

Flowchart



Pseudocode

Start

Input number1
Input number2
Calculate sum
Calculate Average
Display Average

STOP

2. Using selection control structure

- The structure represents choice.
- Execution of statements depends on a condition that returns a true or a false.
- There are four selection controls.

i) If then control

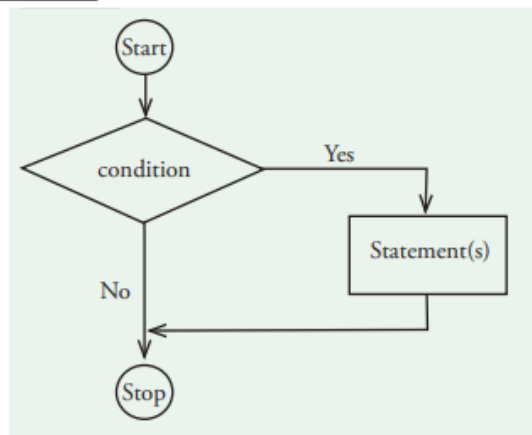
- Used when there is only one option available.

Generalised pseudocode syntax

If Condition **Then** Statement(s)

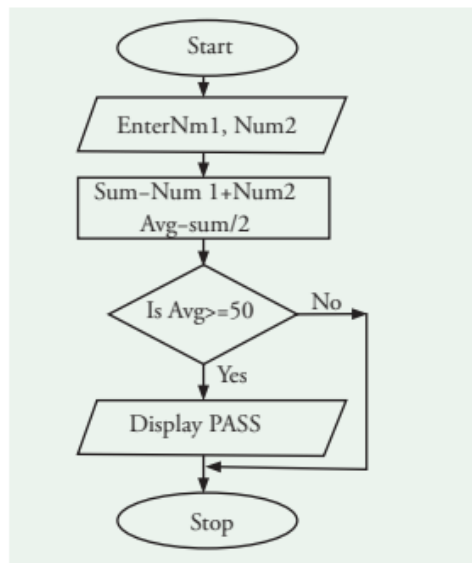
Endif

Flowchart



Example: Draw a program flowchart and write a pseudocode which allows a user to enter marks for two subjects, calculate the total and display "PASS" if average score is 50% and above.

Example: Draw a program flowchart and write a pseudocode which allows a user to enter marks for two subjects, calculate the total and display "PASS" if average score is 50% and above.



Pseudocode

START

Input number1

Input number2

Calculate sum

Calculate Average

IF Avg >= 50 THEN

Display "PASS"

ENDIF

STOP

ii) If then else Control

- Suitable when there are two options to be made.
- When options exceed two the construct is called NESTED IF.

Generalised Pseudocode syntax

If Condition *Then*

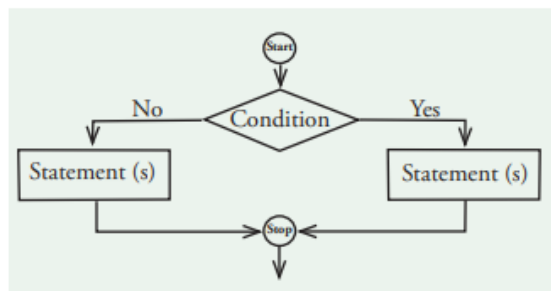
Statement(s)

Else

Statement (s)

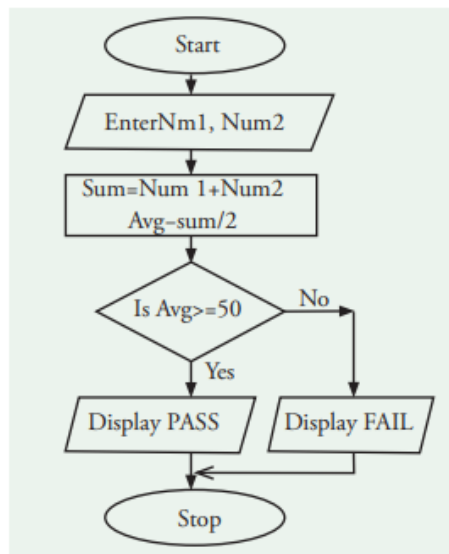
Endif

Flowchart extract



Example: Draw a program flowchart and write a pseudocode which allows a user to enter marks for two subjects, calculate total and display "PASS" if the average score is 50% and above, else "FAIL".

Flowchart



Pseudocode

START

```
Input number1
Input number2
Calculate sum
Calculate Average
IF Avg >= 50 THEN
    Display "PASS"
ELSE
    Display "FAIL"
ENDIF
```

STOP

iii) Nested If

- Used when selection options exceed two.

Generalised pseudocode syntax

```
If condition1 Then
```

```
    Statement1
```

```
Else
```

```
If Condition2 Then
```

```
    Statement2
```

```
Else
```

```
    If Condition n Then
```

```
        Statement n
```

```
    Endif
```

```
Endif
```

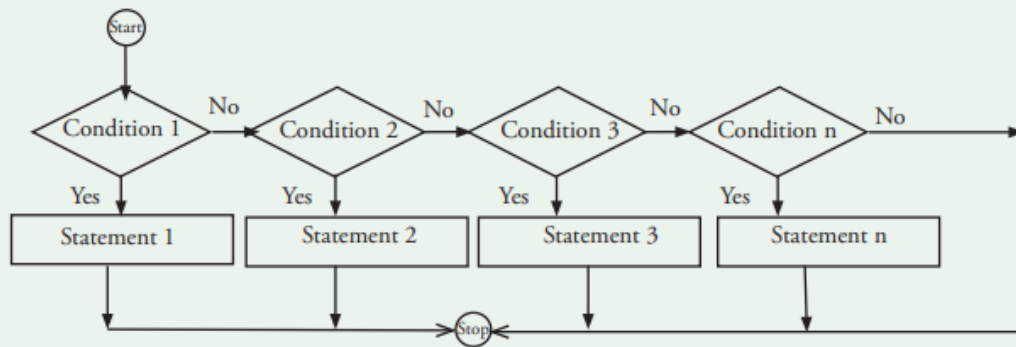
```
.
```

```
.
```

```
.
```

```
Endif
```

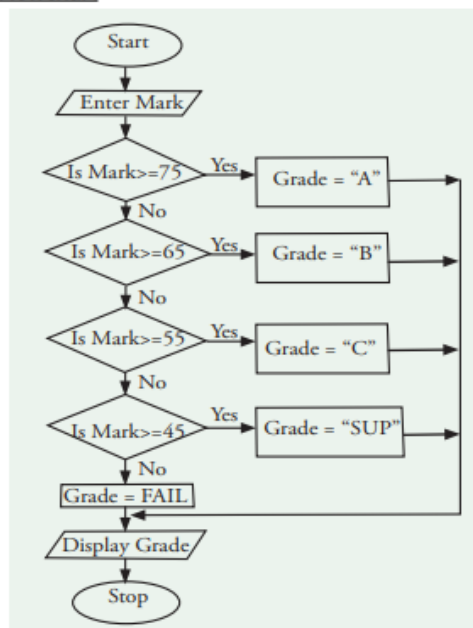
Flowchart extract



Example: Draw a program flowchart and write a pseudocode which allows a user to enter marks and display grades as follows:

Mark $\geq$ 75: Grade is "A"
Mark $\geq$ 65: Grade is "B"
Mark $\geq$ 55: Grade is "C"
Mark $\geq$ 45: Grade is "SUP"
Mark $<$ 45: Grade is "FAIL"

Flowchart



Pseudo code START

```
Input Mark
If Mark >= 75 Then
    Grade = "A"
Elseif Mark >= 65 Then
    Grade = "B"
Elseif Mark >= 55 Then
    Grade = "C"
Elseif Mark >= 45 Then
    Grade = "SUP"
Else
    Grade = "FAIL"
End If
Display Grade
STOP
```

iv) Case control

- Used when there are multiple options.
- Can be used as an alternative for Nested IF.

Generalised pseudo code syntax

Case Condition of

Option 1: Statement 1

Option 2: Statement 2

Option 3: Statement 3

.

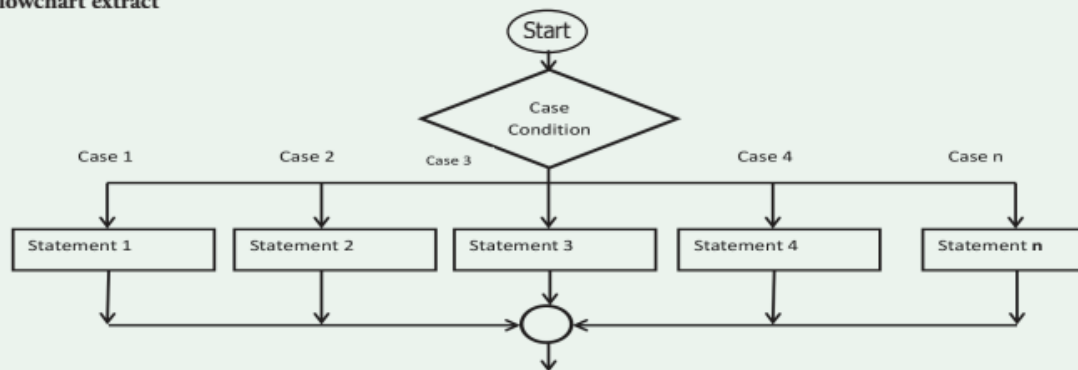
.

.

Option n: Statement n

EndCase

Flowchart extract



Example: Draw a program flowchart and write a pseudo code using CASE statement, which allows a user to enter marks and display grades as follows:

Mark >= 75: Grade is "A"

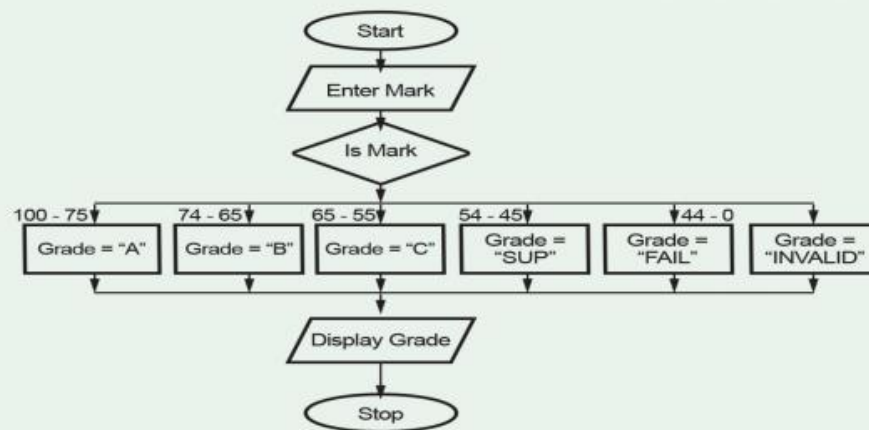
Mark >= 65: Grade is "B"

Mark >= 55: Grade is "C"

Mark >= 45: Grade is "SUP"

Mark < 45: Grade is "FAIL"

Flowchart



Pseudo code

START

Input Mark

SELECT CASE Mark

CASE 100...75: Grade = "A"

CASE 74...65: Grade = "B"

CASE 64...55: Grade = "C"

CASE 54...45: Grade = "SUP"

CASE 44...0: Grade = "FAIL"

CASE ELSE

Grade="Invalid Mark"

END SELECT

Display Grade

STOP

3. Using iteration control structure

- A block of code or statement is repetitively executed as long as a given condition remains true or until a certain condition is fulfilled.

- There are three iteration controls.

These are:

i) While ... Do

The following actions take place when a While...Do loop is encountered.

- The condition is tested.
- If it is true, the loop body is executed and the condition is tested again.
- If the condition is false, While loop terminates and execution continues at statements after the While loop.

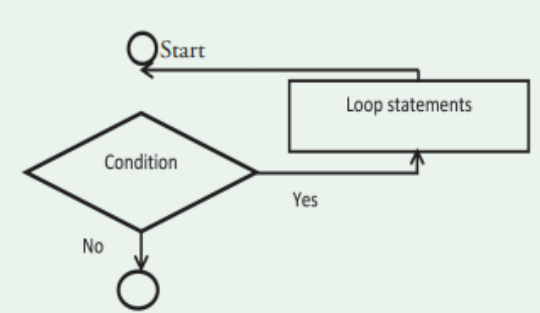
Generalised pseudo code Syntax

While condition

Do Loop statements

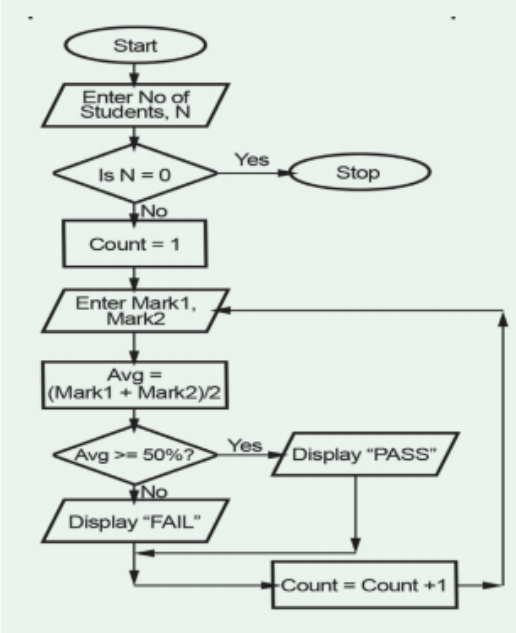
EndWhile

Flowchart extract



Example: Draw a program flowchart and write a pseudocode, which allows a user to enter marks scored in two subjects and display grade of "PASS" if average mark is 50% and above, else "FAIL". This should be for a specific number of students in a class, unless the user enters Zero.

Flowchart



Pseudocode

START

Initialise Counter to 1

REPEAT

Enter NumberOfStudents

Enter Mark1, Mark2

Calculate Average

IF Average $\geq$ 50 THEN

 Display "PASS"

ELSE

 Display "FAIL"

ENDIF

Count = Count + 1

UNTIL N= Count

STOP

Dim number As Integer

number = 1

Do While number $\leq$ 100

number = number + 1

Loop

Ex2

number = 1

While number $\leq$ 100 Do

number = number + 1

end

ii) Repeat ...Until

The following actions take place when the construct is encountered.

- The loop body is executed.
- The exit condition is tested.
- If it is false, the process is repeated from step 1 above.
- When the condition is true, the loop body is exited and control is taken at the point after the condition.

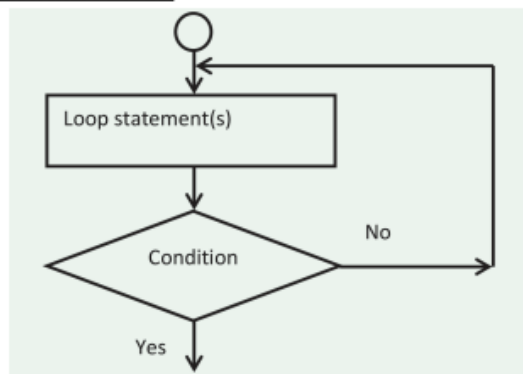
Generalised Pseudocode syntax

Repeat

Loop statement(s)

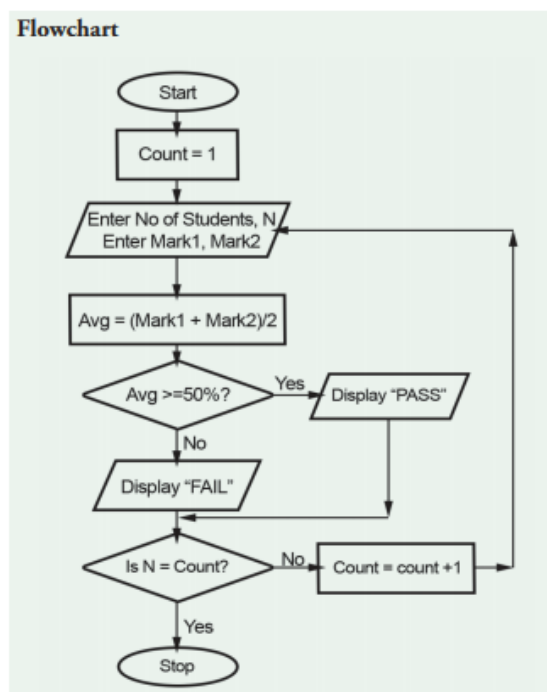
Until Condition

Flowchart extract



Example: Draw a program flowchart and write a pseudo code, which allows a user to enter marks scored in two subjects and display grade of "PASS" if average mark is 50% and above else "FAIL". This should be for a specific number of students in a class.

Flowchart



Pseudocode

```
START Initialise
Counter to 1
REPEAT
Enter NumberOfStudents
Enter Mark1, Mark2
Calculate Average
IF Average >=50 THEN
    Display "PASS"
ELSE
    Display "FAIL"
ENDIF
Count = Count +1
UNTIL N= Count
STOP
```

Comparison between While Do and Repeat Until loops

While ...Do	Repeat ...Until
The condition is tested before the loop body is executed.	The condition is tested after the loop body is executed.
The loop body may never be executed (if the condition is false on the 1st test).	The loop body is always executed at least once.
The loop is exited when the condition is false.	The loop is exited when the condition is true.

iii) For...Do

- Used when certain statement(s) are to be executed for a pre-known number of times.
- Can be used alternatively with While...Do.

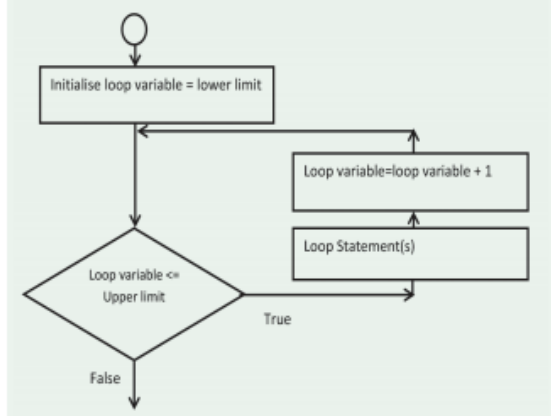
Generalised pseudocode syntax

For loop variable = lower limit to upper limit Do

Statement(s)

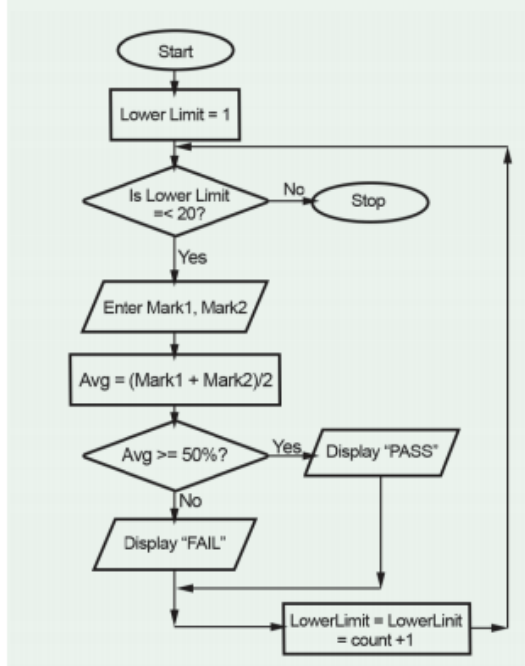
EndFor

Flowchart extract



Example: Draw a program flowchart and write a pseudo code, which allows a user to enter marks scored by 20 students in a class each taking two subjects and display grade of "PASS" if average mark is 50% and above, else "FAIL".

Flowchart



Pseudocode

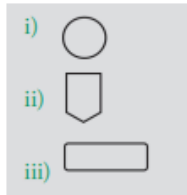
```
START For N = 1 to 20 Do
  Enter NumberOfStudents
  Enter Mark1, Mark2
  Calculate Average
  IF Average >=50 THEN
    Display "PASS"
  ELSE
    Display "FAIL"
  ENDIF
NEXT N
End For
STOP
```

KCSE Sample Questions

1. Distinguish between an assembler and an interpreter as used in programming. (2 marks) **(KCSE 2015)**.
2. State **two** advantages of using low - level programming languages. (2 marks) **(KCSE 2014)**.
3. List two programming language translators. (2 marks) **(KCSE 2013)**.
4. Machine language programs are more difficult to write than high-level language programs. State two reasons for this. (2 marks) **(KCSE 2010)**.
5. Below is a list of program segments in different generations of programming languages. Identify the language for each. (3 marks) **(KCSE 2016)**
 - i) LDA 105
SUB 40
ADD 20
 - ii)

10000110	10111101
01111000	00011100
 - iii) For x: = 1 to 10 do Write (x)
6. List four web programming languages. (2 marks) **(KCSE 2015)**.
7. Give **two** characteristics of scripting languages. (2 marks) **(KCSE 2011)**.
8. State four ways in which a programmer can make a program code easy to follow. (4 marks) **(KCSE 2015)**
9. List two examples of: (2 marks) **(KCSE 2006)**
 - i) Third generation languages
 - ii) Object oriented languages
10. State the stage of program development in which: (2 marks) **(KCSE 2005)**
 - i) A flowchart would be drawn.
 - ii) The programmer would check whether the program does as required.
 - iii) The user guide would be written.
 - iv) The requirements specifications would be written.

11. State the use of each of the following flowchart symbols. (3 marks)



12. a) Below is an algorithm that is used to compute the values of R, S, and T.

P = 5

Q = 6

INPUT N

If N is GREATER OR EQUAL TO 10

R = P * Q

S = Q - P

T = P + Q + R + S

ELSE

R = P + Q

S = Q

T = R + S

END IF

PRINT R, S and T

From the algorithm, determine the output if the input value of N is:

i) 7 (3 marks)

ii) 10 (3 marks)

b) Draw a flowchart for the algorithm in question. (6 marks) (KCSE 2012)

13. In order to process examination results of students in a school, their names, index numbers and scores in 11 subjects are required. The average score for each student is then determined and a grade assigned. This process is repeated for all 40 students in a class.

Draw a flow chart to:

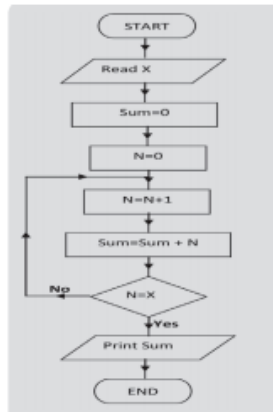
- Read a student's name, index number and the scores in all the subjects.
- Determine the student's average score.
- Assign a grade to the student depending on the average score as follows:

Score	Grade
A	$80 \leq \text{score}$
B	$60 \leq \text{score} < 80$
C	$40 \leq \text{score} < 60$
F	$\text{Score} < 40$

- Display the student's name, index number, average score and the grade.
- Repeat the above steps for all the students in the class. (10 marks) (KCSE 2010).

14. Draw a flowchart to compare three non-equal numeric values A, B, C and print the largest of the three. (11 marks) (KCSE 2006)

15. Figure shows a flowchart. Use it to answer the questions that follow.



i) Determine the output from the flowchart if

a) $X=5$ (2 marks)

b) $X=7$ (2 marks)

ii) Write a pseudocode for the flowchart in figure 3. (5 marks)

iii) Modify the flowchart so that it can be used to get the sum of integers between 50 and 100. (4 marks) (KCSE 2013).