ELEMENTARY COMPUTER PROGRAMING.

Computer Programming
A program – is an organized list of statements (instructions) that when executed, cause the computer to behave in a pre-determined manner or carry out a defined task.

Programming – Refers to the process of developing computer (instructions) programs used to solve a particular task.
A computer program is designed using a particular programming language. Each language has a special sequence or order of writing characters usually referred to as **syntax**

Terms used in programming
Source program.
This refers to the program code that the programmer enters in the program editor window (html editor, Java editor,php(personal home page) editor) that is not yet translated into machine-readable form.

Object code
This refers to the program code that is in machine –readable i.e a source code that has been translated into machine language.

Translators
These are programming tools that translates /convert the source program into object code. E.g. Assemblers, compilers, interpreters etc.

Assembler
An assembler translates a program written in assembly language into machine language.

Interpreter
This translates the source programs line-by-line, allowing the CPU to execute one line before translating the next. The translated line is not stored in the computer memory, hence every time the program is executed, it has to be translated.

Compiler
This translates the entire source program into object code. The compiler translates each high level instruction into several machine code instructions in a process called COMPILATION and produces a complete independent program that can be run by the computer as often as required without the original source program being present.

Levels of programming languages

There are two major levels namely; -
i) Low level languages
ii) High- level languages.

1. Low-level languages.

- These languages are classified as low because they can be directly, or easily understood by the computer with little effort to translate into computer understandable form.
- These languages are hardware oriented and therefore they are not portable. I.e. a program written for one computer cannot be installed and used on other.
-

Types of low level languages

A . Machine language: First generation (languages)

- In this language, instructions are written using binary logic. Given that data and instructions are in binary form, many lines of codes are needed to accomplish even a simple task like adding two numbers i.e. program written in this language look like this.

```
111000110 0000011    10000001
0001111    10001101
10001111   1111111    1000011
```

The program code is hard for humans to understand but it's easily understood by computers.

B. Assembly languages (second generation languages)

- This language is close to the machines vocabulary rather than the human beings vocabulary. It was developed in order to overcome the difficulties of understanding and using machine language. This language helps the programmers to write programs as a set of symbolic operation codes called mnemonics. Mnemonics are basically shortened two or three letter words. A sample program written in Assembly language.

```
Mov   AX, 15        (move 15 to register AX)
SUB   Ax, 10              (subtract 10 from the value Ax.
```

Programs written in this language require an assembler to convert them into machine language.

2. High-level languages.

These languages are very close to the human language (English –like) and they can be read and understood even by people who are not experts in programming. These languages are machine independent. This means that a programmer concentrates on problem solving during a programming session rather than how a machine operates.

Classes of high-level languages

i) Third generation languages (3 GLS)

This generation language is also called structured or procedural languages. A procedural language makes it possible to break a program into components called modules. Each performing a particular task. Structured programming has advantages because it's flexible, easier to read and modify.

Examples of third generation programming language.
Pascal – Was developed to help in teaching and learning of structured programming.
Fortran – Was developed for mathematics , scientists and engineers. It enables writing of programs with mathematical expressions.
Cobol – Was designed for developing programs that solve business programs.
Basic – Developed to enable students to easily learn programming.
c- Used for developing system software e.g the operating systems. Its very powerful high level language because of its ability to provide programmer with powerful aspects / features of low level.
Ada – This language is situatable for developing military, industrial and real time systems.

ii) Forth generation languages (4 GLs)

This generation make programming an even easier task than the third generation language because they present the programmer with more programming tools. Examples of such tools are command buttons, forms etc. the 4 GLs are easy to learn and understand because they are user based. The languages syntax (grammar) is natural , near English language and use menus to prompts to guide a non-specilist or retrieve data with ease.

Examples of 4GLs

a) Visual Basic
b) Delphi Pascal
c) Visual cobol
d) C + +

iii) Fifth generation languages (5 G's)

These languages are designed around the concept of solving problems by enabling the computer to depict human like intelligence. These programs are designed to make the computer solve the problem programmer rather than programmer spending a lot of time to come up with the solution.

Examples of 5GL's
a) PROLOG
b) MERCURY
c) LISP
d) OCCAM.

iv) Object oriented programming languages. (OOP)

The concept behind OOP languages is to look at a program as having various objects instructing to make up a whole. Each object has a specific data

values that are unique to it (called state) and a set of the things it can accomplish called (functions or behavior). This process of having data and functions that operate on the data within an object is called **Encapsulation**. Several objects can then be linked together to form a complete program OOP has greatly contributed to development of **Graphical user interface operating systems and application programs.**

Examples of OOP

a) Java
b) Simula
c) Small talk.
d) Python
e) C++
f) Visual basic.net
g) Sea sharp.

v) Web scripting languages.

These languages are used to develop or add functionalities on web pages. Web pages are hypertext documents created in a language called **Hypertext markup languages** (HTML) .the language consists of markup tags that tell the internet browser that the file contains HTML- code information and is distinguished by a file extension of HTML . the markup tags define the various components of a world wide Web document such as heading , tables , paragraphs , lists etc. HTMl does not have the declaration part and control structures , hence its not considered as a true programming language. Due to its simplist, it has many limitations and can not be used alone when developing functional websites. Some special blocks of codes called Scripts may be inserted in HTML pages using scripting languages like JavaScript, VBScript etc in order to add functionality to HTMl PAGES.

Advantages of Low-level languages.

1. The CPU understands machine language directly without translation.
2. They are suitable and hardly crash or breakdown once written
3. Running a program is fast, no compilation is needed.
4. They are economical in terms of the amount of memory they use.

Disadvantages Low-level languages

1. They are difficult and cumbersome to use and learn
2. Requires highly trained experts to both develop and maintain programs.
3. Debugging programs is difficult
4. They are machine dependant the programs are long.

Advantages of high-level languages

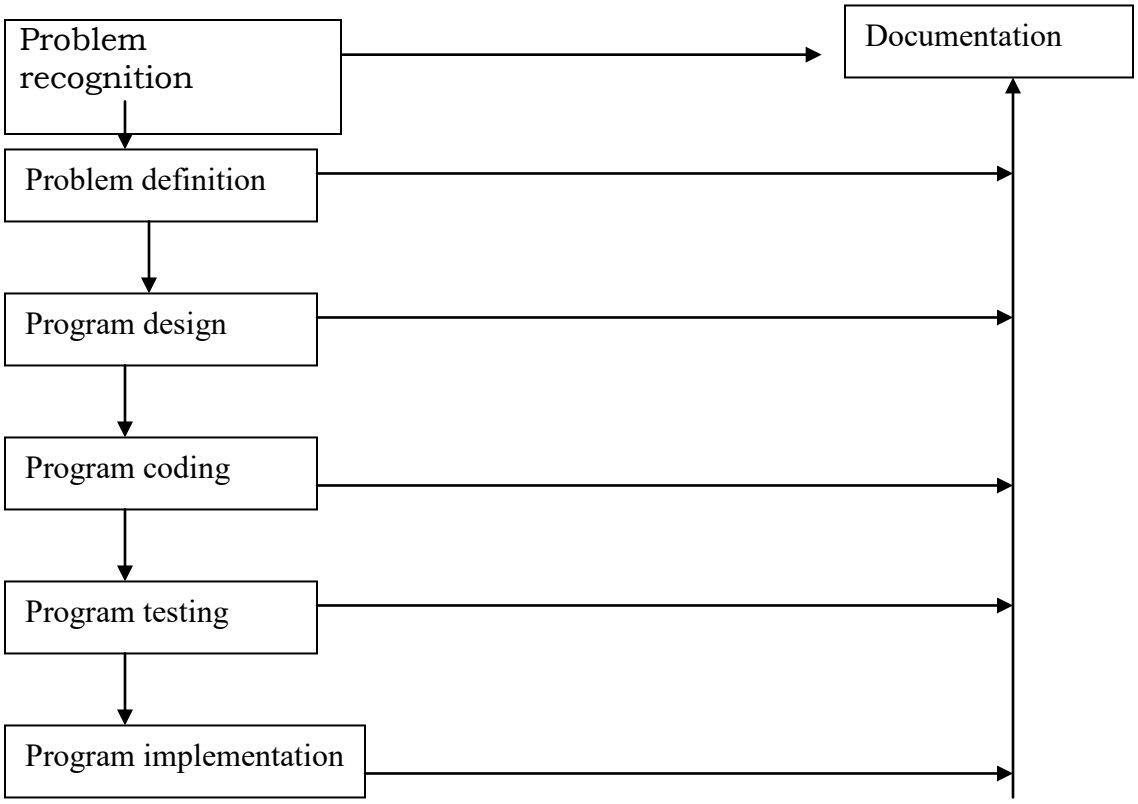
1. The programs are portable (not machine dependant)
2. They are user friendly and easy to use and learn.
3. They are more flexible
1. They provide better documentation
2. They are easy to debug
3. Require less time to code.

Disadvantages of high-level languages

i) Program executed more slowly.
ii) Require larger CPU storage capacity for compilation.
iii)They have to be translated to machine-readable form before the computer can execute them.

Program development

There are six stages of program development. They include



i) Program recognition

This refers to the understanding and interpretation of a particular problem. To understand a problem one has to look for key words such as computer, evaluate, compare etc. a programmer identifies problems in the environment

and seeks to solve them by writing computer program that would provide the solution.

Circumstances that can cause the programmer to identify a problem.

1. Opportunity to improve the current program .
2. Anew directive given by the management requiring a change in the status quo.
3. Problems of undesirable solutions that prevent an individual or organization from achieving their purpose.

Example:- Problem is Finding the area of a circle.
Programmer to develop program used to calculate area of circle.
The equation for calculating the area of circle $A=\pi r^2$

ii) Problem definition.

At this stage the programmer tries to determine or define the likely input, processing and expected output using the key words outlined at the problem recognition stage. The boundaries of the expected program are established and if several methods to solve the same problem are identified the best alternatives should be chosen. At the end of the stage requirement documentation for the new program is written.

iii) Problem design

This is the actual development of the program's processing or problem solving logic called algorithm. (A limited number of logical steps that a program follows in order to solve a problem) the programmer comes up with an algorithm after analyzing the requirements specifications.

Some of the problems are made up of large block code. Ie they are Monolithic while others are made of several units called modules, which work together to form the whole program.

In modular programming each module performs a specific task. This approach makes a program flexible, easier to read and debug. This phase enable the programmer to come up with models of the expected program. The model shows the flow of events and data throughout the entire program from input of a program.

iv) Program coding

This is the actual process of converting a design model into its equivalent program. This is done by creating the program using a particular programming language. The end result of this stage is source programs that can be translated into machine-readable form for the computer to execute and solve the target problem.

v) Program Testing and Debugging.

After coding the program has to be tested and the errors detected and corrected. Debugging refers to detection and correction of errors that may

exist in the program. Program testing involves creating test data designed to produce predictable output.

There are two types of errors (bugs) that can be encountered when testing

i) **Syntax errors** – They occur as a result of improper use of language rules. e.g. grammar mistakes , punctuation , improper naming of the variables etc. These errors are detectable by translator and must be corrected before the program runs.

ii) Logical errors- They are not detectable by the translator. The program rules but gives wrong output or halts during execution.

Methods of error detection

i) Desk checking / dry-run

It involves going through the program while still on a paper before entering it in the program editor.

ii) Using Debugging Utilities.

In the program editor, you can run the debugging utilities during translation to detect syntax errors.

iii) Using Test Data

The programmer carries out trial runs of the new program .At each run he enters various data variation and extremes including data with errors to test whether the system will grid to a halt. A good program should not crash due to incorrect data entry but should inform the user about the anomaly.

vi) Implementation and maintenance Implementation.

This is the actual delivery and installation of the new program ready for use, creating data files and train people to use the system. The new system will change the way things are done hence it should be reviewed and maintained.

vii) Review and maintenance

This stage is important because of the errors that may be encountered after implementation. A program may fail due to poor use, hence proper training and post implementation support of users will reduce chances of having them entering invalid data that crash the program.

viii) Program documentation

This is writing of support materials explaining how the program can be used by users, installed by operators or modified by other programmers. All stages of development should be documented in order to help during future modification of the program.

Types of documentation

i) User oriented documentation

These type enables the user to learn how to use the program as quickly as possible and with little help from grammar.

ii) Operator oriented documentation

Meant for computer operators. E.g Technician. it helps them to install and maintain the program.

Development of algorithms.

Algorithms – Refers to a limited number of logical steps that a program follows in order to solve a problem.

Pseudo code – Refers to a set of statements written in a readable language (English – like) but expressing the processing logic of program.

Guidelines for designing a good pseudo code.

1. The statements must be short, clear and readable.
2. Pseudo code lines should be clearly outlined and indented clearly.
3. It should show clearly the start and stop of executable statements and control structures.
4. Statements must not have more than one meaning.

Examples of Pseudo code.

Write a pseudo code that can be used to prompt the user to enter two numbers, calculate the sum and average of the two numbers and then display the output on the screen.

Solution

START
Print “Enter two numbers”
Input x,y
Sum – x+y
Average = sum /2
PRINT sum
PRINT Average
STOP

Program flowcharts.

A flowchart is a diagrammatic representation of a program in algorithms. It uses statements and symbols that have specific meaning. The symbols are of different standard shapes linked to show the order of processing. Each shape contains note stating what the operation is.

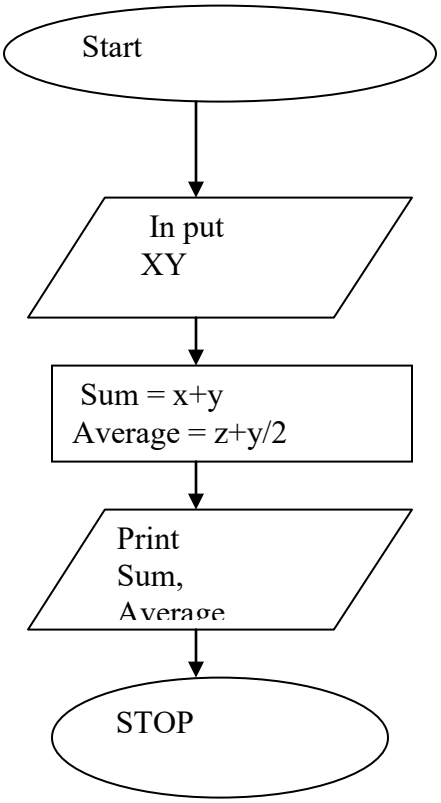
Guidelines for drawing a flowchart

1. There should be only one entry and one exit point of a program algorithm.
2. Use correct symbol at each stage in the flowchart.
3. Avoid a crossd flow lines.
4. Be as neat and tidy in drawing as possible.
5. Genearal direction of flow in any flowchart is from top to bottom , left to right.

Examples of flowchart

Draw a flowchart for a program used to prompt the user to enter two number s. the

program should find the sum, and average then display the output



Types of flowchart

System flowchart
It's a chart that depicts the systems as a whole with only subsystems or a major elements shown.

Program flowchart
This chart shows the sequence of operations as carried out by a computer program.
Advantages of flowchart.
1. Gives programmer good visual reference of what the program will do.
2. Serves as program or system documentation.
3. Its easy to trace through from the start to find the action resulting from a set of condition .
4.Allows programmer to test alternative solutions to a problem without over coding the program.

Disadvantages.
1. There are so many different ways to draw them.
2. Its difficult to remain neat and uncluttered if the logic is complex.

1. Constructing a flowchart is time consuming.
2. They take up considerable space.
3. They are difficult to amend without redrawing

Program control structure
They are blocks of statements that determine how statements are to be executed.
There are 3 control structures namely.
i) Sequence
In this control structure, the computer reads instructions from a program file starting from the first top line and proceeding downwards one by one to the end. Hence sequential program execution enables the computer to perform tasks that are arranged consecutively one after another in the code.

Examples of how a sequential program execute

```
Begin {procedure name}
        Action 1
        Action 2
        Action n
End     {procedure name}
```

the program file reader reads sequentially statements by statements to the end of the file.

ii) Selection/decision
This structure is used to branch, depending on whether the condition returns a value of True or False (yes or no)

For example

```
If < condition >
        Then Action 1
        Else Action 2
Endif.
```

There are 4 types of selection controls used in high-level programming
1. IF……..THEN
This is used if only one option is available. All other options are ignored. For example if a school wants to reward only the students who have mean mark of 80 and above, it will be reward only those students who have attained 80% and above and ignore the rest.

General format

```
If < condition > Then
Statements :
Endif
```

```
If mark >80 then
Print “Reward”
Endif
```

IF……THEN……ELSE
Used when there are two available options for example in a football match a player is given a RED CARD if he does a very serious mistake otherwise he is given Yellow Card.

General Format
```
If < condition >THEN
        Statements 1,
ELSE
        Statement 2
EndIF
```

The algorithm will be;-
```
If fault = serious THEN
        Print "RED CARD"
ELSE
        Print " yellow card"
EndIf.
```

3. Nested IF
This is used where two or more options have to be considered to make a selection. For example, to award grade according to the marks as follows

a) 80 marks Grade A
b) 60 marks Grade B
c) 50 marks Grade C
d) 40 marks Grade D

General format
```
If < conditions >Then
        Statement
ELSE
If < condition >Then
        Statement
ELSE
If < condition >Then
        Statement
ELSE
        Statement
        EndIf
        EndIf
        End
```
CASE SELECTION
Its an alternative to the nested IF. This selection is preffered to the Nested if in order to reduce the many lines of codes . case selection can only be expressed using integers and alphabetic characters only. The Boolean expression should be CASE interger OF or CASE char OF.

General format
CASE x of
Label 1: statement 1
Label 2: statement 2
:
label n: statement n-1
ELSE
Statement n
End case

Example

CASE	average OF
80…100:	Grade = “A”
70-79	Grade= “B”
60-69	Grade = “C”
40-49	Grade =” E”
ELSE	
	Grade = “F”

End case.

iii) Iteration (coping) repetition

This is designed to execute the same block of code again and again until a certain condition is fulfilled . Itelaration is important in situations where the same operation has to be carried out on a set of data many times.

There are three main looping controls.

i) THE – WHILE-DO LOOP

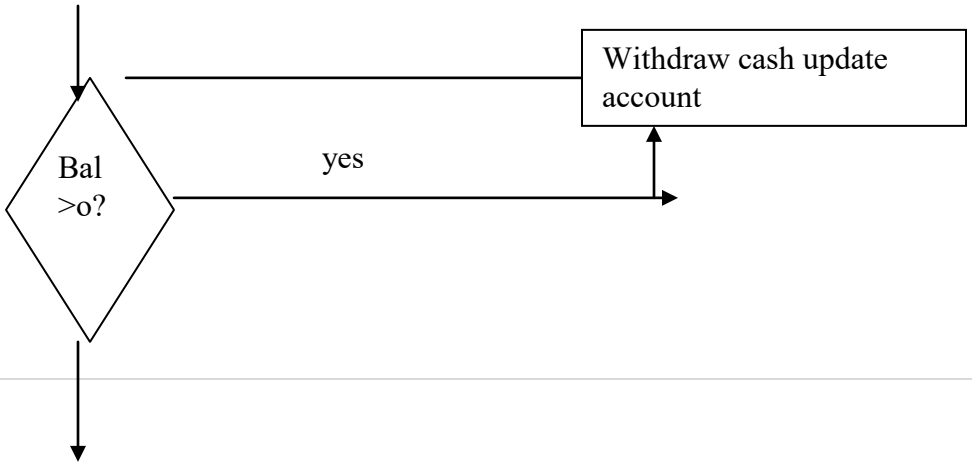
This repetitive structure tests the condition first before executing the successful code if and only if the Boolean expression returns a true value.

Example

To withdraw money using an ATM a customer must have a balance in his/her account.

General format
While <condition >Do
Statement
End while.

Flowchart



End loop

no

Pseudo code segment
While balance >O do
Withdraw cash
Update account
End while

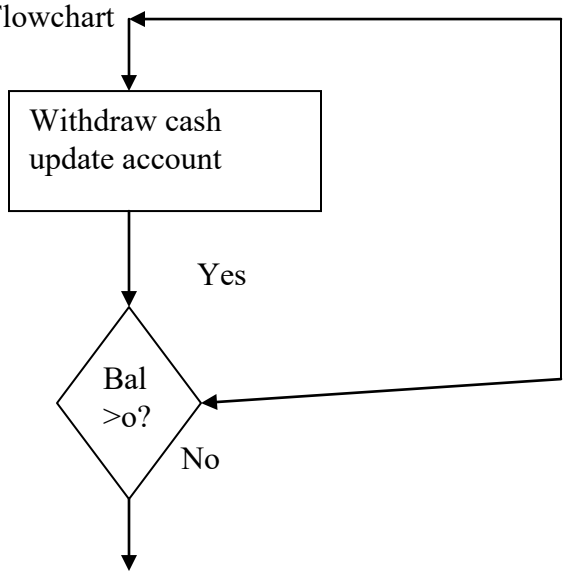
Example
Pseudo code segment
REPEAT
Withdraw cash
Update account
Until balance ≤0:

General format
REPEAT
Statement
UNTIL <condition>

iii) REPEAT……UNTIL LOOP

In this structure the code is executed before testing the condition. The repeat loop stops when the Boolean statement returns a value. For example in the case of ATM of discussed above the client can withdraw money until balance is zero.

Flowchart



Exit loop

iii) The FOR LOOP

This structure is used where execution of the chosen statements has to be repeated a predetermined number of times. For example if a program is to calculate the sum of ten numbers provided by the user. The FOR LOOP can be used to prompt the user to enter the 10 numbers at most ten times. Once the numbers are entered the program calculates and displays the sum.

Pseudo code

```
FOR count = 1 to 10 Do
Writeln "Enter a number (N)"
Readln N
Sum = sum +N
End FOR.

Display sum
```

The counter has to be set to a start value and sometimes to an end value.

General fomart of the loop

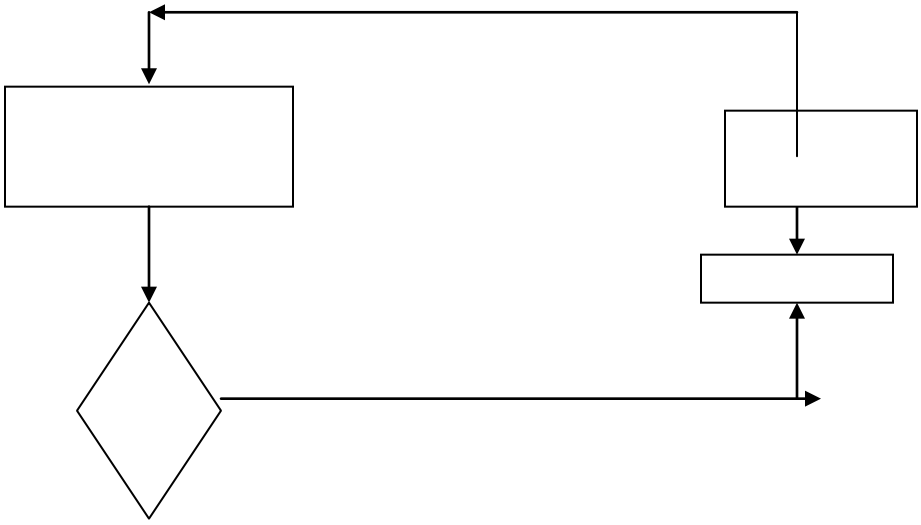
1. Fomart for the FOR loop that counts from lower limit.

```
For loop variables = lower limit To upper limit Do
        Statements
        Endfor
```

2. Fomart for the "for" loop that counts from upper limit down to lower limit .

```
for loop variable = Upper limit Down To lower Limit Do
        Statements
        Endfor
```

Flowchart for a forloop that counts upwards.



Flowchart for a FORLOOP that counts downwards

Diagram

Examples of complex pseudo codes.

1. Unshirika society pays 5% interest on shares exceeding 10,000 Ksh and 3% on share that do not meet theis target. However no interests is paid on deposits in the members bank. Account . Design a pseudo code for a program that would:

a) Prompt the user for shares and deposits of a particular member.
b) Calculate the interest and total savings.
c) Display the interest and total savings on the screen for a particular member.

Pseudo code
Start
Input Name, share,Deposit
If share> 10000 THEN
Intrest = 0.05 x shares
ELSE
Interest = 0.03 x shares
EndIf
Total savings = Deposit Interest + shares
Print Name., Totalsaving , Interest
Stop
Flowchart
Diagram
2. Botswana has a population of 3,000,000 but thi si filling by 4% each year . The island of Reunion has a population of 95,000 but this is increasing by 95 each year.
a) Draw a flowchart that predicts the year in which the population of reunion will be grater than that of Botswana if the trend continues.
Solution
Pseudo code
Start
Bts: 3,000
IR: = 95,000
Year : = 0
REPEAT
Bts = Bts – (Bts * (4/100)
IR = IR + (IR (9/100))
Year = year +1
UNTIL IR > Bts
Print year
Stop

3. Write a program that will allow the input of name of student marks obtained in 5 subjects (Math, Engllish, Computer, Biology) .
The program should calculate the total and evearge marks for each student and assign the grades depending on the average marks obtained as follows.
80- 100 A
70- 79 B
60- 69 C
50-59 D
Below 50 –E
The program should then display each students name , total marks and average .

Pseudo code
START
REPEAT
Print " Enter name and subject marks"
Input Name, maths,English , Kiswahil , Computer , Biology
Sum = maths,English , Kiswahil , Computer , Biology
AVG = sum/5
If (AVG ≥ 80) AND (AVG ≤ 100) THEN
Grade = "A"
If (AVG ≥ 70) AND (AVG ≤79) THEN
Grade = "B"
If (AVG ≥60) AND (AVG ≤69 THEN
Grade = "c"
If (AVG ≥ 50) AND (AVG ≤59) THEN
Grade = "D"
ELSE
Grade = "E"
Endif
Endif
Endif
Endif
Print name, sum, AVG, Grade
Until count = Number of students.
Stop
Flowchart

PAST KCSE QUESTIONS ON THE TOPIC
1. 2002
State two types of documentation in program development and give the purpose of each .
(4 marks)
2. state any three activities that occur in a program compilation process (3 marks)
3. The following can be used to list the add numbers between 0 and 100

1. Diagram
a) write a program segment for the flowchart using a high language (7 marks)
b) What would be the output from the flowchart if the statement in the decision box is changed to
(3 marks)

i) odd = 100 ii) odd <100 iii) odd >100

2003
1 a) Distinguish between Machine and Assembly language (2 marks)
b) State the type of translator necessary for a program written in
i) High level language
ii) Assembly language
2. Briefly explain the purpose of the following types of a program documentation (2marks)

i) User manual
ii) Reference guide

3. State any two features of user-friendly program (2 marks)

2. Study the flowchart below and the question that follow.

a) Write a high level language program for the above flowchart (7 marks)

b) List the outputs of the flow chart above (5 marks)

KCSE 2004

1. Distinguish between a compiler and an interpreter (2 marks)

2. What is meant by the term DRY Running as used in program development (2 marks)

3. Differentiate between source program and object program (2 marks)

4. Bidii wholesaler has two categories of customers for order processing. Category "A" obtains 10% discount on all orders up to Ksh 10,000. Otherwise the discount is 20% on the entire order. Category "B" obtains 30% discount on all orders if the debt repayment is "good" otherwise the discount is 15% . Draw a flowchart for the order processing (15 marks)

KCSE 2005

1. Distinguish between Real, Integer and character data types a used in programming (3 marks)

Diagram

2. a) Name the control structure depicted by the flowchart above (1 mark)

b) Explain the following terms as used in program implementation (2 marks)

i) Parallel running
ii) Direct changeover

3. a) State the stages of program development in which (2 marks)

i) A flowchart would be drawn
ii) The program would check whether the program does as required.
iii) The user guide would be written
iv) The requirements specification would be written

b) State the output of the following segment.

Diagram

c) Draw a flowchart to computer the combined resistance ® of two resitors R1 and R2 in parallel using the formula: (5 marks)

$$R = \frac{1}{\frac{1}{R1} + \frac{1}{R2}}$$

KCSE 2006

1. a) List two examples of

i) Third generation language
ii) Object oriented languages.

2 . 2007

Write al algorithm to compute the area of a triangle (2 marks)

PRACTICE QUESTIONS ON THE TOPIC

1. Distinguish between the following

a) Compiler and interpreter.

b) Object code and source code.

2. State 3 advantages of high level languages over low level language.
3. Outline the stages of program development in their respective order
2. State two advantages of modula programming.
3. distinguish between pseudo code and Algorithm.
4. explain three types of control structures used in programming
5. write a pseudo code that will inform the user of what to waer depending on the weather . if its raining “wear rain coat” if not “wear overcoat”
6. draw a flowchart for a program to display the name of a suspect to a crime who is aged between 20 and 35 year and between 66 and 70 inches tall.
7. draw a flowchart to compute and print the grades for an examination. The input Data is Roll. No and marks for six subjects out of 100 . grades are allocated on the following basis

% marks	grades
75 and above	A
60 and less than 75	B
Less than 60	C

PREDICTION QUESTIONS ON THE TOPIC

1. a) What is meant by structured programming (1 mark)

b) State 3 advantages of using modules in program development (2 marks)

2. Give a reason why its necessary to have a program design (1 mark)
2. Distinguish between user documentation and operator documentation (2 marks)
3. state two advantages and two disadvantages of using flowchart in program design (4 marks)
4. using a simple sketch , illustrate the
    i) REPEAT…….UNTIL control structure (3 marks)
    ii) WHILE ……..DO control structure
    iii) CASE control structure
5. Give an advantage of compiling a program rather than interpreting it (1 mark)