COMPUTER SOFTWARE/COMPUTER INTELLIGENCE

- Computer software refers to ***sets of electronic instructions***, ***commands*** or ***programs*** which direct the computer on the capture, processing, storage and outputting of information.
- ***Software*** defines a ***computers intelligence*** – what a computer can do or used for. Computers work on ***programmed instructions*** as provided by the ***programmer/software engineer.*** Hence computers having ***artificial intelligence.***

Differences between computer hardware and computer software

Computer Hardware

- It is physical/tangible
- Hardware defines a computers presence

Computer Software

- It is intangible. They are internal instructions that tell computers to do whatever they do
- They define performance of a computer.
- Software has got power

Categories of Software

Computer software is classified basing on; the element of the computer system that directly uses it. Elements of the computer system include:- users, hardware, software and procedures.

Hence;

1. Application software
2. System software.

Application software

- All softwares a computer user directly interacts with to do work on and with the computer. It is user dependent.
- Application software takes two forms. That is;
  - **Customized** or **in-house** or **tailor made** or **user application** or **Bespoke application software.**
  - **Packaged/off-the-shelve /standardized application software**

Customized or **in-house** or **tailor made** or **user application** or **Bespoke software**

- Refers to all softwares designed to be used by one organisation
- Customized software can be in form of;
    - **Vertical customized application software** – i.e. if it is used in one department or section of the organisation.
    - **Horizontal customized application software** – i.e. if it can be used in several or other departments and sections of the company.

An evaluation of customized softwares

Advantages of customized apps

- Hard to manipulate.
- Makes the organisation using it look unique. Improve image.
- Customized softwares applications offer very high productivity.
- Usually caters for all organizational needs by providing for all the necessary details.

Disadvantages of customized apps

- Usually very expensive to design and develop. All production costs are met by one organisation.
- They are not flexible. Since they can only be used by one company.
- They take long time to develop.
- They need a lot of specialized skills to use since they are not very common.

Packaged/off-the-shelves /Standardized application software

These are copyrighted and commercial softwares designed to meet software needs of a wide variety of users.

Packaged softwares can be;

- **Generalized packaged apps –** if they can be used for many tasks like some word processors
- **Specialized packaged apps:** if they handle tasks of a specialized nature like accounting and marketing apps.

Advantages of off-the-shelve software

- They are always available in software kiosks.
- They are cheap to procure.
- They are very flexible since they can be used by any organization.
- They are the core of entertainment and leisure application programs.
- Do not need a lot of specialized skills.
- They form the majority of educational reference software.

Disadvantages of off-the-shelve software

- They are very easy to manipulate since they are almost known by many people
- They are not very secure.
- May not handle some problem with ease. May not provide the necessary software needs breadth and depth.

Software acquisition

- Softwares are generally distributed through;
  - Software writers retail outlets in major cities around the world,
  - Software writers agents
  - Software writers certified firms.
  - **Application service providers (ASP):** Third party organizations that manage and distribute software and other computer services online (through the WWW).

Other forms of software and concepts

- ***A Cross-platform application*:** Any software that runs identically on multiple operating systems. Different platforms like Windows, Linux and Mac
- ***Copyrighted software:*** Authentic software with copyright owners or true software writer.
- ***Pirated software:*** Stolen software without copyright restrictions. It is typical of tampered security features.
- ***Freeware***. This is copyrighted software provided at no cost for users. Usually provided by government agencies and other donors.
- ***Shareware***. Is copyrighted software that is distributed for free during its trial period, but payment is required for its continued usage beyond the trial period.

Other forms of software and concepts

- **Public Domain software**. This is free software donated for public use and has no copyrighted restrictions.
- **Software suite**: A collection of applications with common basic interface features sold as a single unit e.g. Microsoft Office and Lotus SmartSuite.
- ***Software version:*** A major upgrade in a software product – Office 2003 and Office 2007.
- ***Software release:*** A minor upgrade in a software product – Office 2007 and Office 2010.
- Software perch: Updates provided by the software writer periodically.

General Common types of application software

- Word Processing software
- Spread sheet software
- Database software
- Presentation software
- Integrated software
- Computer aided design software
- Desktop publishing software
- Project management software – Ms. Project
- Personal information managers
- Video and Audio editing software
- Multimedia authorizing software
- Web page authoring software
- Personal finance software
- Educational Reference software
- Entertainment software
- Communications software
- Accounting software
- Pointing and Image editing software

SYSTEM SOFTWARE

These are programs or softwares a computer uses to manage itself. System softwares provide ***control***, ***coordination***, ***planning*** and ***reporting*** of system activities.

System softwares are to the computer system as application softwares are to the computer user.

They include;

- The operating system
- Utility or service programs
- Programming tools
- System drivers
- Firmware

The Operating System

- Is a master control program that manages the ***general operations*** and ***resources*** of the computer system. For example;
  - **Android**
  - **Windows family OS like Windows XP, 7, 8, 10, (Windows Millennium)Me, 2000, NT, 95, 3.1, 98, etc**
  - **Mac**
  - **Linux**
  - **Unix**
  - **Disc Operating system (DOS)**
  - **Wang**

Classifications of OS

1. **Multi–tasking OS:** This allows many tasks to be handled by a single CPU system at the a time. ***Multi-tasking*** Operating Systems are ***multi-processing*** and ***multi-user*** Operating Systems
2. **Single user OS:** These are operating systems that allow only one activity in the CPU at a time.

Functions of the operating system in the computer system

1. **Manage storage media/allocates memory**: They determine on where data and programs should be placed at the time of processing and after processing.
2. **Management of computer system security:** Enable computer users to create user accounts with passwords.
3. ***Manages system prompts***: They direct or prompt devices and programs to start work.
4. ***Facilitate the booting process:*** Loading the OS into memory is one of the last processes of the booting process. Without a working OS the computer cannot boot. Booting is the process of attaining the operational run-time environment of the computer system. The booting process can be;
    1. ***Cold/hard booting:*** Starting the PC when the system has been off.
    2. ***Warm/soft booting:*** Restarting the PC when the OS is already running in memory

Fun**ctions of the OS** Cont'd

5. **Management of system faults and errors:** The OS keeps on checking on system devices and programs. Where it finds an error it reports it to the user or make an effort to fix it. **They monitor system performance and usage**.
6. ***Provide system user interface:*** The OS provides tools and mechanisms (interface) through which the user interacts with the computer. The OS provides;
    I. **Menu driven interface – where users interact through lists of options they choose from.**
    II. **Command line interface through command boxes where commands are punched.**
    III. **Voice recognition interface**
    IV. **Touch and wave interface**
    V. **Remote terminal or controlled interface**

Fun**ctions of the OS** Cont'd

VI. Graphical user interface (*GUI*): The OS provides interactive ***dialogue boxes***, ***buttons***, ***icons***, and ***tabs*** with which the user can work with the computer.

Advantages of GUI

- **Graphical images are easier to learn and work with.**
- **There is no need to type and memorize any hard command language.**
- **The interface is similar for any application.**
- **They make program identification easy. Each program has got a unique graphical image**

Fun**ctions of the OS** Cont'd

VI. **Device configuration**. The OS initiates or aligns devices such that they interact well with the entire computer system. Configuration also refers to the alignment or arrangement of the functional part of the computer system.

VII. They provide a platform or foundation into which application programs run.

Fun**ctions of the OS** Cont'd

Disadvantages of GUI

- Graphical images require faster memory and faster processor because they generally heavy.
- It also occupies more disk space to hold all files for different functions.
- It is difficult to automate functions for expert users.

SYSTEM UTILITIES

- A utility program is a form of system software that ***services*** other programs and system devices, enhance system performance or make the computer system more user friendly and cost effective.
- They are also called **Service programs**.
- Some utility programs come ***embedded*** in operating systems, while others can be bought independently or down loaded online.

Classifications of utility programs.

Popular categories and types of utility programs include:

1. File management utilities like;
   - File sort utilities: rearranges files in either descending or ascending order of date, type, name or size.
   - File viewer: Copies and enables files to be copied and viewed.
   - File compression utility. Reduces or compresses the size of a file for more storage space & faster movement on a network.
   - Diagnostic utility. compiles technical report about programs and devices

1. File management utilities like Cont'd;
   - ***Backup utility:*** Enables computer users to create extra copies of the same file for reference purposes
   - ***Un installer:*** enables users to remove no-longer useful or contaminated programs.
   - ***Text editors:*** Enables creation and editing of text based files using basic text formatting features. They do not work with graphics or images.
   - ***Recycle bin:*** Contains/holds deleted files and folders
   - ***Disc cleanup:*** Enables the user to clear unwanted files and folders.

Classifications of utility programs cont'd

2. Malware/duty data management utilities

- ***Antivirus utility:*** They detect, prevent and remove viruses from a computer memory or storage media.
- ***Anti-spyware utilities:*** Prevent system activities from being spied on by third parties.

3. Software and device optimizers or enhancement utilities

- **Language translators like compilers and interpreters**
- ***Disk defragmenter:*** reorganizes files and also ensure that bad disc sectors and files are isolated for enhanced performance.
- ***System restore:*** It restores the system to a known previous state that worked well.
- **Calculators and Calendar utilities**

Classifications of utility programs cont'd

4. Device maintenance utilities

- ***Scanner Disk:*** It check system disks or storage media for faults/errors and attempts fixing them where possible. ***They*** identify and correct storage media (hard disk) physical and logical errors
- ***Screen saver:*** It causes the monitor/screen to display a moving image or blank screen if no keyboard or mouse activity occurs for a specified time period.

Programming tools

Definitions

Programming tools are sets of system programs used to create other programs. They include;

- **Programming languages**
- **Language translators (like interpreters, compilers and assemblers)**
- **Debugging utilities**
- **Linkers**

Programming language. They are platforms or system programs used to create other programs or softwares

NB:

Some programming languages are sold separately from other programming tools, while others are sold as a package consolidated into one pack called an ***Integrated Development Environment*** (IDE)

Classifications of programming languages

- ***Low level:*** They are programming platforms where programmer use data binary codes or specific abbreviations when developing programs or software.
    - **Machine language or 1st generation languages**
    - **Assembly languages or second generation languages**
- **High level languages: They are programming platforms where programmers use natural words, statements and objects in the program development process. E.g.** C, C++, Pascal, Fortran, BASIC, Java, Lisp, Smalltalk.

High level languages:

(i) **BASIC** (Beginning All purpose symbolic Instruction Code). This was developed in 1964 by John Kemeny and Thomas Kutz to teach students how to use computers.

(ii) **FORTRAN**. (FORmula TRANslation). This was developed in 1956 to provide an easier way of writing scientific and engineering applications

High level languages:

(iii)COBOL (Common Business Oriented Language). Developed for developing business application programs

(iv)PASCAL. Was developed in early 1970 specifically for computer scientists.

Others:

- **ADA**
- **ALGOL**
- **PL/M**
- (Programming Language Microcomputer)
- **LOGO**

OBJECT ORIENTED PROGRAMMING Languages (OOPL)

- OOPL Use objects to represent data and behavior (like motion). Examples include Visual **Basic (C++)**

Basic programming terms

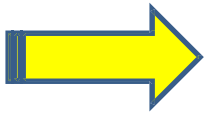
- **Code**: Is a written program text or statement a computer can execute

NB: Computers understand a binary language where each character (letter, number, shape, light or sound stream)has got its binary code of 8bits or byte

- **Source code:** Is a program code in a language a computer programmer understands.
- ***Object code:*** Is program code which is computer–readable. It is a code in a language which is understandable by the computer i.e. a source code that has been translated into machine understandable language.
- ***Translator:*** It is a program or utility that interprets (translates) a program code from high level to low level language.

Program or language translators

Language translators are categorized as;

1. ***Interpreters*:** These utilities translate program code text file or data statement by statement.
2. ***Compilers:*** They translate the entire program code (text file or data) at once. 
3. ***Assembler***: An assembler translates a program written in 2nd generation (assembly) language into machine language or first generation language

NB: In the process of coding and compilation two types of errors are committed as explained below;

Program errors

1. **Syntax errors/procedural errors**: These errors occur as a result of improper use of language rules. e.g. grammar mistakes , punctuation , improper naming of the variables

Forms of syntax errors

- **Program grammar errors**
- **Spelling error *for instance - writing a keyword with wrong spelling***
- **Punctuation errors or poor punctuation for example - m*issing semicolon to terminate execution of line of code***
- **Missing Parenthesis e.g. (}) to indicate the start and end of main function.**
- **Printing the value of variable without declaring it**
- **Using a function that is not in the included header**
- **Using wrong case for keywords**
- **Poor use of space as a character**

2. Logical errors: These errors are not detectable by the translator. The program runs but gives a wrong output.

Error detection methods/approaches

- **Dry run/desk check** – through the program script
- **Error detection utilities** – like debugging utilities
- **Use of test data** – giving the program sample data to manipulate.

Basic terminologies

- **Algorithm:** Refers to a limited number of logical steps that a program follows to solve a problem. Program algorithm can be represented by;
    - **Pseudo code**
    - **Flow chart**
- ***Pseudo code:*** Refers to a set of statements written in a natural readable language (like Luganda, English or Japadhola, Lukiga, etc) but expressing the processing logic of program. It is a set of statements written in a readable language (English – like) but expressing the processing logic of program.

Sample Pseudo code: A pseudo code for adding and averaging two numbers

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

Flow Chart

A flowchart is a diagrammatic representation of a program's algorithm

Method 2: Pseudocode

Begin

Set π to $\frac{22}{7}$

Prompt the user for the radius (R).

Store the radius in a variable

(R)

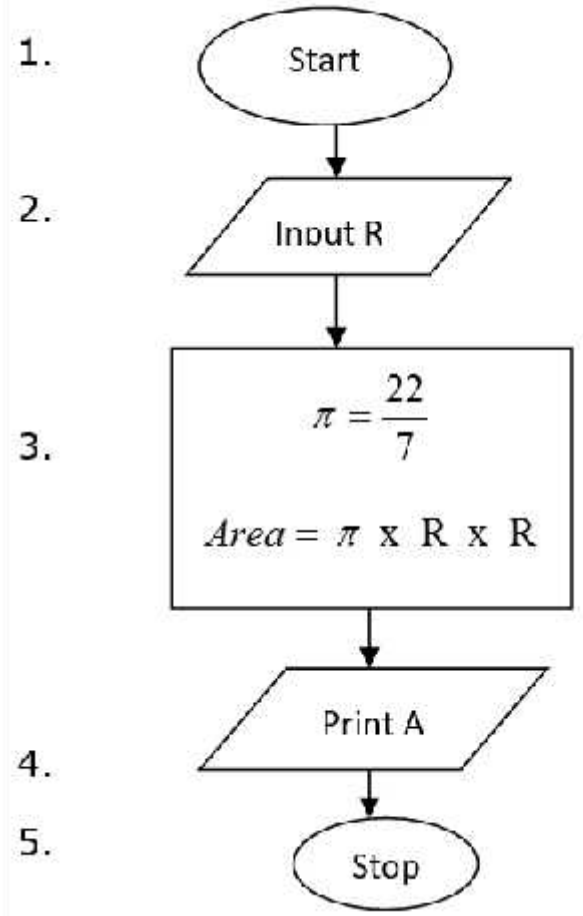
Set area (A) to π x R x R

Store the area in a variable (A)

Print A.

Stop

Method 1: Flowchart



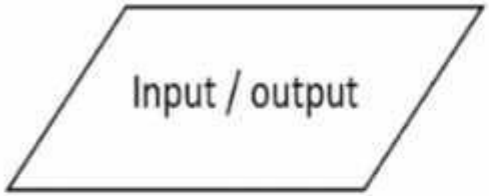
Flow chart construction

- When constructing a flow chart we use both statements and special symbols with specific meaning
- The symbols are combined with short text clues which are easily understood by programmers

1. 

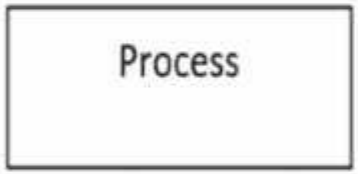
Ellipse:

Denotes the beginning and end of the program algorithm.

2. 

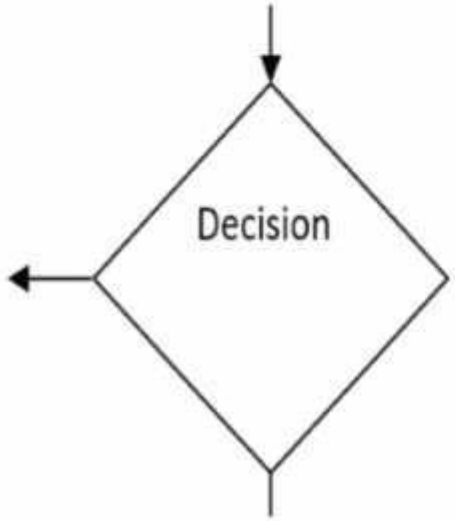
Parallelogram:

Used to denote an input or output operation. For example, READ A, B, PRINT SUM.

3. 

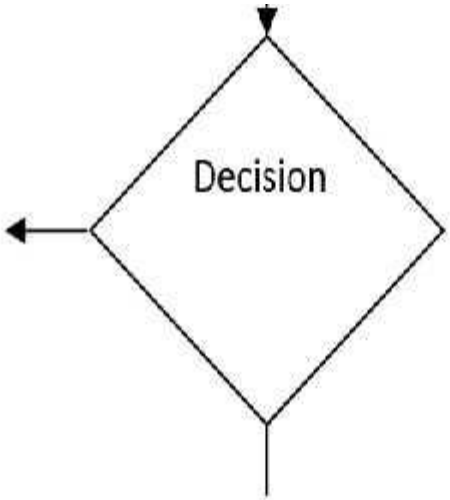
Rectangle:

Indicates that a processing or data transformation is taking place. For example SUM = A + B.

4. 

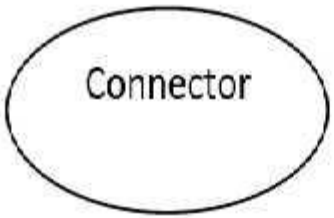
Rhombus:

Used to specify a condition. A condition must evaluate to a *Boolean value* (True or false) for the program to execute the next instructions.

4. 

Rhombus:

Used to specify a condition. A condition must evaluate to a *Boolean value* (True or false) for the program to execute the next instructions.

5. 

Connector:

Used as a connecting point or interface for arrows coming from different directions.

6. 

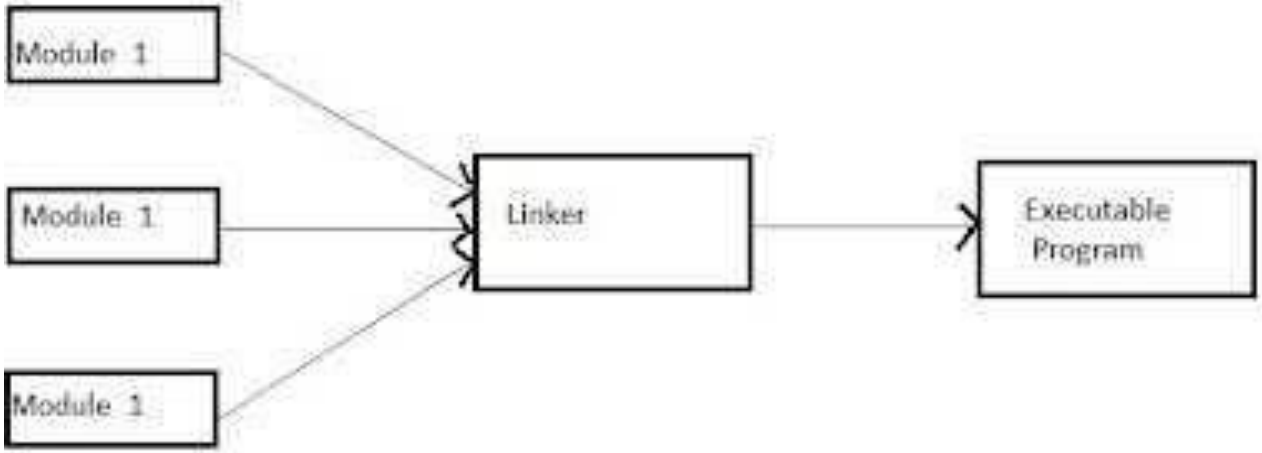
Arrow:

Used to indicate the direction of flow of the program logic.

A linker

- A program that "pulls" other sub-programs together in order for a program to work or run as a single unit. It combines several separate program modules into one program unit. It helps in resolving internal differences between the different program modules.
- ***Linking*** is a process of joining object code file to all the other files that make a full executable program

WINWORD.exe is a linker for Ms. Word Program.



Loader:

A Loader is a programming utility that sets up an executable program into main memory for execution. It loads the machine codes into memory.

Loading is the final sage of the compiling / assembly process.

A variable

- A name given to some memory location that stores values subject to change.

For instance

Int x, y (at the declaration of variables level)

A constant

- Is a fixed value a program may not alter during execution.

x = 2

y = 5

2 and 5 are constants ***at the definition of variables level***

A function

A ***function*** is a group of statements that together perform a task. They are usually in or preceded by round brackets

For instance;

int main () – (as the main function)

A statement is a collection of expressions that usually end with a semi-colon

printf("David is handsome");

Getchar();

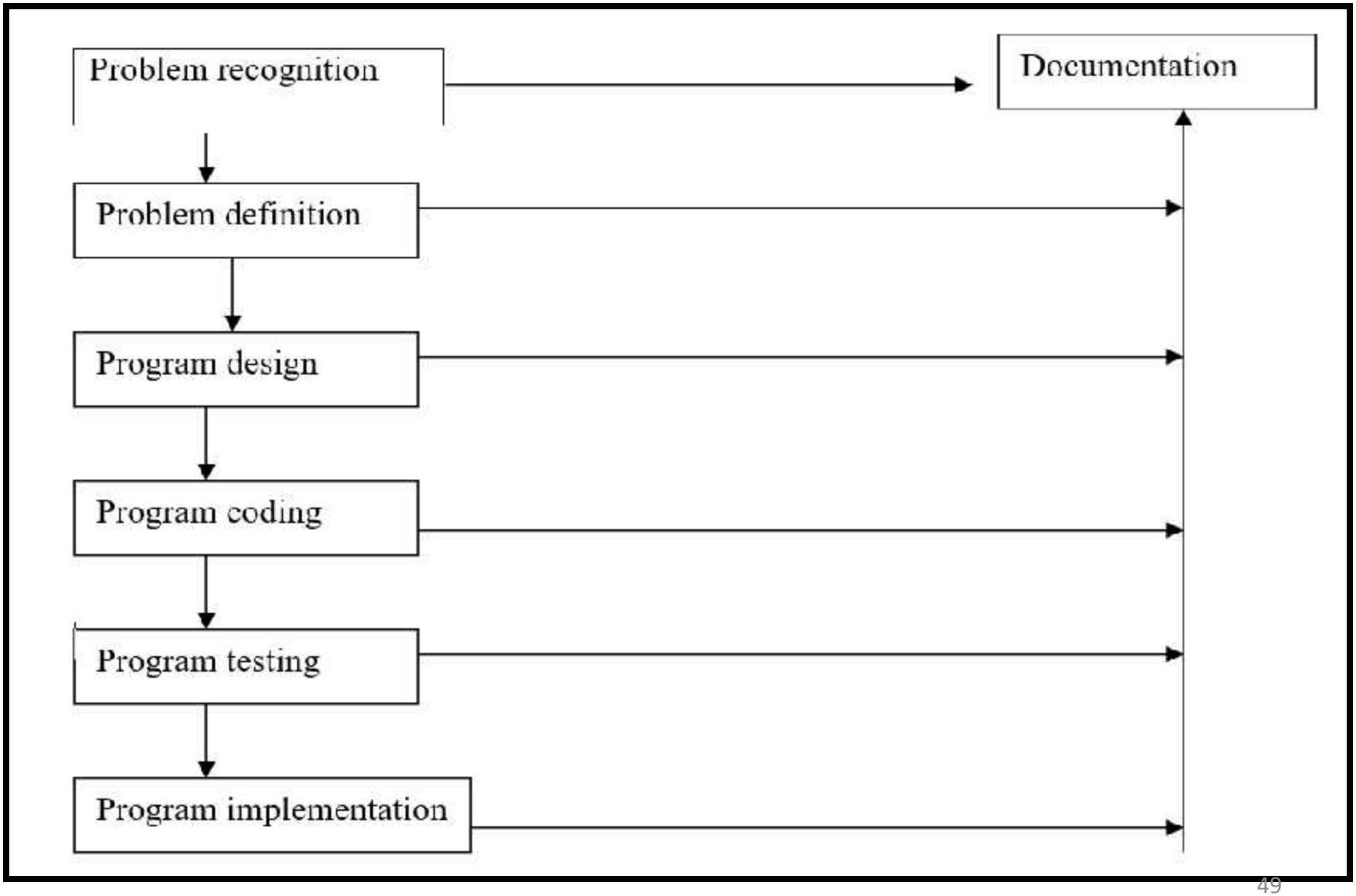
Return0 ();

Program Development Cycle

Software developers go through a number of activities/stages or processes before coming up with the software we use. The most common stages are;

1. Problem recognition/Identification
2. Problem definition/Analysis
3. Program design
4. Program coding
5. Program testing and debugging
6. Implementation and maintenance
7. Program Documentation

Diagram showing program Development Stages.



1. Problem Recognition

- Identify the problem in the environment ….

1. **Problems** or undesirable situations that prevent an individual or organization from achieving their purpose
2. **Opportunity** to improve the current program.
3. **A new directive** given by the management/Authorities requiring a change in the status quo

Example: A Student has received his examination papers from 3 exams in a subject. He is having trouble arriving at his average grades for the term.

2. Problem Definition/Analysis

…Determine or define the likely …

1. Input (Given Data)
2. Processing activities (How?)
3. The expected output (Required Data)

…Write out a requirements report/document

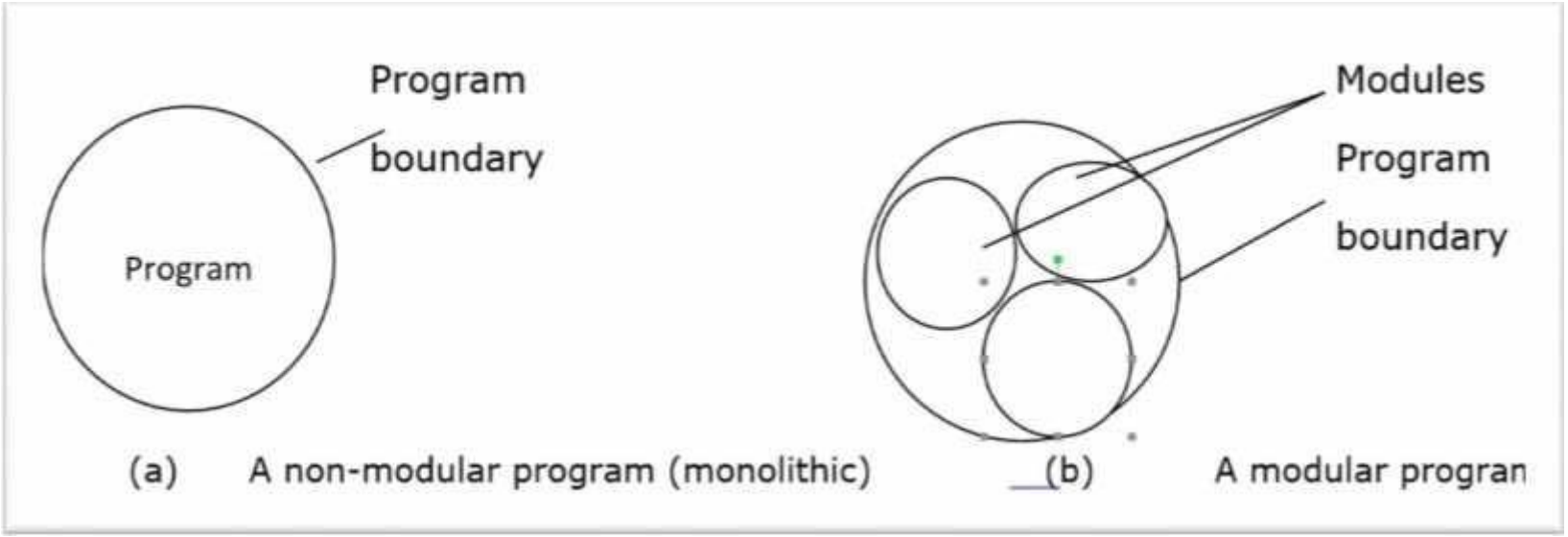
3. Program Design

- ..is the actual development of the program's processing or problem solving logic called the **algorithm**
- Algorithm = a definite number of **logical steps** that a program follows in order to solve a problem
- …Result is a Pseudo code / Flow chart
- Monolithic Vs Modular

Monolithic = One large block of code

Monolithic Vs Modular Program Design

In modular programming, each module performs a specific task. This approach makes a program flexible, easier to read and carry out error correction.



4. Program Coding

- ..is the actual process of converting a design model into its equivalent program
- ..is done by typing using particular programming language application software.
- .. end result is a source program (source code) that can be translated into machine readable form for the computer to execute and solve the target problem

5. Program Testing/Debugging

- the program is tested in order to detect & correct errors (debug)
- **Logical Errors**

… errors that are not detectable by the translator. The program runs but gives wrong output or halts during execution.

6. Implementation and Maintenance

- **Implementation** = actual delivery and installation of the new program ready for use
- **Review and maintenance =** is important because of the errors that may be encountered after the program has been implemented or exposed to extensive use

++ A program may also fail not because of poor development but also due to poor use

- Proper training
- Post implementation support of users

7. Program Documentation

- Writing down formal support materials
- Internal Vs external documentation

1. **User oriented** = how to use the program
2. **Operator Oriented** = How to install and maintain program
3. **Programmer orientated** = for skilled programmers for future modification

Qualities of a good programming language

- Suitability of the problem.
- Clarity and simplicity.
- Efficiency – well organized and detailed enough
- Availability
- Consistency – stable and reliable
- Economy – it must fit in the budget

Considerations when buying a programming platform/language

- Cost
- Nature of program to develop
- Experience
- Availability
- Consistency – stable and reliable
- Compatibility with available hardware and software
- Source

Revision Questions - 1

1. Mention two forms software
2. Name three categories of system software giving an example in each
3. Give two examples of application programs
4. Distinguish between custom made and off shelf application packages
5. Distinguish between open source and proprietary software
6. State three advantages of graphical user interface over command line interface

Revision Questions - 1

7. Differentiate machine language and high level languages
8. Define a bug in programming
9. Differentiate between program and programming language
10. Briefly explain of following terms in programming language.
    a) **Source code**
    b) C**ompiler**
    c) **Assembler**
    d) **Interpreter**
    e) **Linker**

Revision Questions - 1

11. Explain three qualities of a good programming language
12. Write computer program code you know using a programing platform of your choice
13. What is a device driver?
14. State two uses of device drivers to a computer user?
15. What is a software suite?
16. Give three advantages of buying/using software suites compared to buying single software applications.

Review Questions for the topic

17. Give one advantage of compiling a program rather than interpreting it.
18. Outline at least six stages of program development in their respective order.
19. Highlight two disadvantages of monolithic programs.
20. State two advantages of modular programming.
21. In what stage of the development does program documentation fall? Justify your answer.
22. Differentiate between a flowchart and pseudocode.
23. What is a program bug?
24. Explain why it is important to test a program before implementing it.
25. State three characteristics of computer software