FACULTY OF PHYSICAL SCIENCE
NNAMDI AZIKIWE UNIVERSITY AWKA

Effective leadership is not about making mouth watering promises and speeches or being liked ; Leadership is defined by **results** not attributes

VOTE

Chukwuma Stanley C.

FOR **PRESIDENT**

FACULTY OF PHYSICAL SCIENCE
STUDENTS ASSOCIATION (FOPSSA)

#StandWithStan

LETS REFORM FOPSSA TOGETHER

- Vacuum tubes
- Transistor
- Integral Circuit
- micro processor
- Artificial intelligence

...ches, a much faster computer than the Mark I was made possible by John ...uchly and J. Presper Eckert at the University of Pennsylvania. They built the ...IAC (Electronic Numerical Integrator and Calculator) for the U.S. government to ...erate gunnery tables for use in the Second World War. Not finished until 1946, ...IAC contained 18,000 vacuum tubes, weighed over 30 tons, occupied 1600 square ...et of floor space, and required 100 kilowatts of power. Heat dissipation was a ...oblem and the vacuum tubes were not very reliable; when one burned out, it had to ...e replaced. This computer was programmed by rewiring control boards to do the ...sired functions. The ENIAC was much faster than the MARK I, capable of ...thmetic operations in fractions of a second. It was said at the time that this one ...achine would keep all the world's mathematicians busy for over 200 years!

...e difficulty that the ENIAC and MARK I had was that their design used decimal ...presentation of numbers. John von Neumann, based on an idea of Claude Shannon, ...posed that binary numbers would simplify design of computers since binary ...mbers require only two states to represent numbers. This gave rise to The EDSAC ...lectronic Delay Storage Automatic Calculator) which was built in England in 1949 ...ing binary circuits and binary representation of control programs and data. This was ...uickly followed by the EDVAC (Electronic Discrete Variable Automatic Computer).

...ommercial applications soon followed the development of these machines. ...NIVAC, an analog computer was one of them. It was used in the census of 1952 ...nd attracted much attention when it was used on television to predict the election of ...resident Eisenhower. These commercial machines were very expensive. They used ...acuum tubes that were costly, required large amounts of power, gave off ...considerable heat, and had finite life times. Nonetheless, because of their accuracy ...and speed, the number of computers increased exponentially; by 1953 over 2,000 ...were in use.

...In 1956 the transistor was invented at Bell Labs. Smaller, cheaper to manufacture, ...requiring less power, and more reliable; it quickly replaced the vacuum tube in ...computers. With the improvements in the computer itself, improvements in peripheral ...devices such as card readers were also witnessed. Instead of punched cards, magnetic ...tape was introduced for the storage of information and programs. Small regions of iron on the tape could be magnetized or not to represent 1's and 0's of the binary code for numbers and information. Where one punch card could only hold 80 characters, many thousands of characters could be packed into several inches of magnetic tape.

By 1965, further developments were made in the computer world and computers incorporating *integrated circuits* were introduced. Integrated circuits are small chips of silicon wafer containing many small transistors and circuits packed into the size of a postage stamp. Smaller, cheaper, and more reliable than even the transistor, they made the computers of this era faster, smaller, cheaper, and more powerful. The trend continues and computers incorporating *Large Scale Integration or Very Large Scale Integration of circuits* were developed. These improvements have made possible personal computers, ones cheap enough to be owned by individuals as well as companies. The next phase of computers are expected to be based on Artificial Intelligence, ones that will recognize sound and sight and will be able to *learn* and be *taught*.

1.3 Classifications of Computers

Computers can be classified in many ways and these classifications depend on their functions and definitions. They can be classified according to purpose, type of data they handle, generation, size and storage capacity.

1.3.0 Classification according to purpose

According to purpose, computers are either general purpose or special purpose.

- *General purpose computers*: they are designed to perform a range of tasks. By using a general purpose computer and different software, various tasks can be accomplished, including writing and editing (word processing), manipulating facts in a data base, tracking manufacturing inventory, making scientific calculations, or even controlling organization's security system. In fact, virtually all computers from micro to mainframe are general purpose. They have the ability to store numerous programs, but lack in speed and efficiency.
- *Special purpose computers:* these are designed to handle a specific problem or to perform a specific task. While a special purpose computer may have many of the same features found in a general purpose computer, its applicability to a particular problem is a function of its design rather than to a stored program. The

A PDA

They are used for Web browsing, office applications, watching videos, viewing photos or as mobile phones. PDA model features vary, but current common features include touch screen displays, Bluetooth and Wi-Fi connectivity, memory card slots, mobile software applications and multimedia support. PDAs usually include personal information managers for contacts and schedules and always come with software to synchronize desktop or cloud server information.

- *Smart phone*: this is a cellular phone that performs many of the functions of a personal computer, typically having a touch screen interface, Internet access, and an operating system capable of running downloaded applications. Smart phones are compact in size and often only slightly bigger than standard mobile telephones.

Lenovo A5000 smart phone

1.3.3 Classification according to generation

Each generation of computer is characterized by a major technological development that basically changed the way computers operate, resulting in increasingly smaller, cheaper, and more powerful, efficient and reliable devices.

First Generation (1940-1956): the first generation computers used vacuum tubes for circuitry and magnetic drums for memory, and were often enormous, taking up entire rooms. They were very expensive to operate and in addition to using a great deal of

electricity, generated a lot of heat, which was often the cause of its malfunctions. They were hard wired and operated in millisecond speed.

First generation computers relied on machine language, the lowest-level programming language understood by computers to perform operations, and they could only solve one problem at a time. Input was based on punched cards and paper tape, and output was displayed on printouts. Examples include UNIVAC (Universal Automatic Computer), ENIAC (Electronic Numerical Integrator and computer) and EDSAC (Electronic Discrete Variable Automatic Computer)

Second Generation (1956-1963): transistors replaced vacuum tubes to usher in the second generation of computers. The transistor was invented in 1947 but did not see widespread use in computers until the late 1950s. The transistor was far superior to the vacuum tube, allowing computers to become smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation predecessors. Though the transistor still generated a great deal of heat that subjected the computer to damage, it was a vast improvement over the vacuum tube.

Second-generation computers still relied on punched cards for input and printouts for output. They however, moved from cryptic binary machine language to symbolic (assembly) language, which allowed programmers to specify instructions in words. High-level programming languages were also being developed at this time, such as early versions of COBOL and FORTRAN. These were also the first computers that stored their instructions in their memory moving from a magnetic drum to magnetic core technology. Again, the computational time of Second generation computers reduced to microseconds from milliseconds. Examples: Honeywell 400, IBM 7030, UNIVAC LARC, CDC 1604

Third Generation (1964-1971): the development of the integrated circuit (IC) was the hallmark of the third generation of computers. An IC contained only about ten to twenty components (Small Scale Integration (SSI)).

Instead of punched cards and printouts, users interacted with third generation computers through keyboards and monitors and interfaced with an operating system, which allowed the device to run many different applications at one time with a central program that monitored the memory. Computers for the first time became accessible

Converting to base 8: here we divide by 8

54/8 = 6 R 6 ↑
6/8 = 0 R 6 |

Therefore 54_{10} is equivalent to 66_8.

Converting to base 2: Since the base to be converted to is 2; we divide by 2

54/2 = 27 R 0 ↑
27/2 = 13 R 1 |
13/2 = 6 R 1 |
6/2 = 3 R 0 |
3/2 = 1 R 1 |
1/2 = 0 R 1 |

The result is 110110_2 (in reverse order)

Converting to base 16: to convert to base 16, we divide by 16

54/16 = 3 R 6 ↑
3/16 = 0 R 3 |

Therefore 54_{10} is equivalent to 36_{16}.

Example 3.4.2: Convert 84.25_{10} to binary, octal and hexadecimal.

This example contains fraction i.e. there is a decimal point and digits after the decimal point. In this scenario, we group the number into two: the Left Hand Side (LHS) and the Right Hand Side (RHS). LHS is the whole number part and RHS is the fractional part. To convert the LHS, repeated division by the destination base is performed as in example 3.4.1. However, the RHS is converted by multiplying the digits by the base; keeping the whole number resulting from the multiplication and

repeatedly multiplying the remainder by the base until a remainder of zero is obtained.

Solution

Converting to base 2: the LHS is 84, this we divide by 2

84/2 = 42 R 0 ↑
42/2 = 21 R 0 |
21/2 = 10 R 1 |
10/2 = 5 R 0 |
5/2 = 2 R 1 |
2/2 = 1 R 0 |
1/2 = 0 R1 |

giving 1010100_2

The RHS is .25, we then multiply by 2 as follows

.25 x 2 = 0.5 ↓
.5 x 2 = 1.0 ↓

producing .01

Thus, 84.25_{10} is equivalent to 1010100.01_2

Converting 84.25_{10} base 8: again, divide the LHS by 8 and repeatedly multiply RHS by 8

84/8 = 10 R 4 ↑
10/8 = 1 R 2 |
1/8 = 0 R 1 |

we obtain 124_8

for the RHS

.25 x 8 = 2.0

Therefore, 84.25_{10} is equivalent to 124.2_8

ii. We perform direct conversion by grouping the numbers in three's.

Example 3.6.1: Convert 1101011_2 to octal

Step I: convert to 1101011 to decimal

N is 7, N-1 = 6;

$1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 1 \times 64 + 1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

$= 64 + 32 + 8 + 2 + 1$

$= 107_{10}$

Then convert 107_{10} to octal

107/8 = 13 R 3 ↑

13/8 = 1 R 5 |

1/8 = 0 R 1 |

This will yield 153_8

Therefore, 1101011_2 is equivalent to 153_8

Example 3.6.2 : convert 10111.11_2 to base 8

Solution N = 5, N-1 = 4;

$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

$= 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 + 1 \times 0.5 + 1 \times 0.25$

$= 16 + 0 + 4 + 2 + 1 + 0.5 + 0.25$

$= 23.75_{10}$

We again convert 23.75 to base 8;

For the LHS, we have

23/8 = 2 R 7 ↑

2/8 = 0 R 2 |

Giving us 27_8

For the RHS, we multiply by the base which is 8

Thus $.75_{10}$ to X_8 is

$.75 \times 8 = 6$ R 0

We then combine the LHS and the RHS to obtain $(27.6)_8$.

Method 2:

Example 3.6.3: convert 1101011_2 to octal by direct conversion

Since 2^3 is 8; the possible number of combinations is as shown in table 3.0

Octal	Binary equivalent		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 3.0: possible number of combinations for 2^3

To convert, we first make sure that the number of digits in the number to be converted is divisible by three. If the number of digits is not divisible by three, we append 0's either to the LHS or to the RHS without changing the value of the number in question. For example, the number of digits in example 3.6.3 above is 7; which is not divisible 3; we then append 0's to the left to obtain:

0011101011_2 i.e. (1101011 ≡ 001101011)

Step 2: group in 3's to obtain

001 101 011

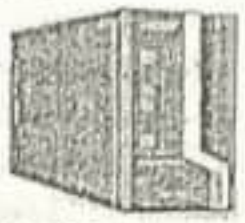
Step 3: from the table, find the equivalent of the digits in octal

001 = 1, 101 = 5, 011 = 3

Thus $1101011_2 = 153_8$

Mini Computers

Minicomputers are much smaller and less expensive than mainframe computers. They possess most of the features found on mainframe computers, but on a more limited scale. Mini computers can still have many terminals (from 4 to about 200) but not as many as the mainframes.



Acer Veriton 6620G: (mini computer)

They can store a tremendous amount of information, but again usually not as much as the mainframe. Medium and small businesses typically use these computers. Examples include PDP-1, IBM AS/400, IBM SYSTEM 360, VERITON 6620G, HP 3000 and PRIME 9755.

Micro Computers

A microcomputer or personal computer (pc) is a small, relatively inexpensive computer that is available to only one user at a time. The central processing unit of the microcomputer is embedded in a single chip (*microprocessor*). They are used in schools, homes and offices. Microcomputers can be grouped into the following:

- *Desktop:* this is a pc that is not designed for portability. It is expected to be set up at a permanent place.

A desktop computer

- *Workstation:* this is a desktop computer with more powerful processor, additional memory and enhanced capabilities for performing its tasks.

- *Notebook:* this is also called *laptop*. It is a portable computer that integrates the display, keyboard, a pointing device or trackball, processor and memory in a battery-operated package slightly larger than an average hardcover notebook.

Notebook pc

- *Tablet pc*: tablet PCs are wireless, portable personal computer with a touch screen interface. Popular uses for a tablet PC include viewing presentations, video-conferencing, reading e-books, watching movies, sharing photos and more. Tablet PCs run on a variety of operating systems, like Android Jellybean (an open-source OS built by Google) and Windows 8 which experiences dramatic increase in popularity.

Tablet pc: MICROSOFT Surface Pro 3

While it is possible to type directly on the surface of a tablet, some users prefer a wireless or Bluetooth-connected keyboard. More tablets are now offered with optional docking stations with keyboards thereby transforming the tablet into a full-featured notebook. All tablets include some form of internal storage but if you find yourself needing more, most models are able to have additional space added for a relatively low price.

- *A personal digital assistant (PDA)*: this is a portable device that functions as a personal information manager.

Example 3.6.4 Convert 10111.11_2 to base 8

Solution

Step 1: make the number of digits divisible by 3

This will produce

010111.110

Step 2: group in 3's

010 111 110

Step 3: from the table,

010 = 2, 111 = 7 and 110 = 6

Therefore, $10111.11_2 = 27.6_8$

3.8 Conversion from binary to hexadecimal

Conversion from binary to hexadecimal can again be done in two ways:

i. We first convert the binary number to decimal and then from decimal to hexadecimal. i.e

$$X_2 \longrightarrow X_{10} \longrightarrow X_{16}$$

ii. We perform direct conversion by grouping the numbers in four's.

Example 3.7.1 convert 11011011_2 to base 16

Method 1:

Step 1: convert 11011011_2 to decimal; N = 8, N-1 = 7;

$1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 1 \times 128 + 1 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

$= 128 + 64 + 0 + 16 + 8 + 0 + 2 + 1$

$= 219_{10}$

Step 2: convert 219_{10} to base 16

219/16 = D R B ↑

D/16 = D R D |

producing DB_{16}.

Therefore, 11011011_2 to hexadecimal = DB_{16}

Example 3.7.2: Convert 1110111011011.10101_2 to base 16 by direct conversion

Since 2^4 is 16; the possible number of combinations is as shown

hexadecimal	Binary equivalent			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Table 3.1: possible number of combinations for 2^4

To convert, we first make sure that the number of digits in the number to be converted is divisible by four. If the number of digits is not divisible by four, we append 0's either to the LHS or to the RHS without changing the value of the number in question.

For the LHS, the number of digits is 13 which is not divisible by 4, so we append zero to obtain:

0001110111011011

And for the RHS, we have 1010100

Step 2: group in 4's to obtain

0001 1101 1101 1011 . 1010 1000

Step 3: from the table, find the equivalent of the digits in hexadecimal

0001 = 1, 1101 = D , 1101 = D , 1010 = A and 1000 = 8

Thus $1110111011011.10101_2 = 1DDA8_{16}$

Converting to base 16: similarly divide the LHS by 16 and multiply the RHS by 16

84/16 = 5 R 4 ↑

5/16 = 0 R 5 | Giving 54_{16}

For the RHS,

.25 x 16 = 4.0

Thus, 84.25_{10} is equivalent to 54.4_{16}

3.6 Conversion from other bases to base ten (decimal)

To convert from other bases to decimal, we first establish the number of digits in the LHS of the number, say N. We then subtract 1 from it to obtain N-1; the weight of the MSD. Multiply the digits by their corresponding weight; after which the LHS is added to the RHS to obtain the result.

For example, consider the number 6527. The number of digits (N) is 4, then N-1 is 3. Thus the MSD 6 has a weight of $(base)^3$. To convert 6527_8 to decimal, we have

$6 \times 8^3 + 5 \times 8^2 + 2 \times 8^1 + 7 \times 8^0$
$= 6 \times 512 + 5 \times 64 + 2 \times 8 + 7 \times 1$
$= 3072 + 320 + 16 + 7$
$= 3415_{10}$

Note: *X^0 is 1 (law of indices)*

Example 3.5.1: convert 77.34 in octal to decimal

Solution: LHS is 77, N is 2 and N-1 is 1; thus 77.34_8 to decimal is gotten as follows:

$7 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} + 4 \times 8^{-2}$
$= 7 \times 8 + 7 \times 1 + 3 \times 0.125 + 4 \times 0.015625$
$= 56 + 7 + 0.375 + 0.0625$
$= 63.4375_{10}$

Example 3.5.2: convert 10110101 in binary to decimal

Solution

N = 8, N-1 = 7. Then 10110101_2 to X_{10} is as follows:

$1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 1 \times 128 + 0 \times 64 + 1 \times 32 + 1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$
$= 128 + 0 + 32 + 16 + 0 + 4 + 0 + 1$
$= 181_{10}$

Example 3.5.3: convert 11101.101_2 to X_{10}

Solution

N = 5, N-1 = 4.
$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$
$= 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 + 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125$
$= 16 + 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125$
$= 29.625_{10}$

Example 3.5.4: convert $45AD_{16}$ to X_{10}

Solution

N = 4, N-1 = 3; then $45AD_{16}$ to decimal is as follows:
$4 \times 16^3 + 5 \times 16^2 + A \times 16^1 + D \times 16^0$
$= 4 \times 4096 + 5 \times 256 + 10 \times 16 + 13 \times 1$
$= 16384 + 1280 + 160 + 13$
$= 17837_{10}$

Example 3.5.5: convert F9B.1A in hexadecimal to decimal

Solution:

N = 3, N-1 = 2, then $F9B.1A_{16}$ to decimal is converted as follows:
$F \times 16^2 + 9 \times 16^1 + B \times 16^0 + 1 \times 16^{-1} + A \times 16^{-2}$
$= 15 \times 256 + 9 \times 16 + 11 \times 1 + 1 \times 0.0625 + 10 \times 0.0039$
$= 3840 + 144 + 11 + 0.0625 + 0.039$
$= 3995.1015_{10}$

Conversion from binary to octal

Conversion from binary to octal can be done in two ways:

We first convert the binary number to decimal and then from decimal to octal. i.e.

$X_2 \longrightarrow X_{10} \longrightarrow X_8$

include: Troubleshooting or diagnostic programs, Antivirus programs, Uninstall programs, Backup programs and File compression programs.

Device drivers: these are specialized programs that allow communication between a device and the computer. A device driver must be installed whenever a new device is added and is loaded into memory each time a computer is started.

Language translators: a computer language translator is a program that translates a set of code written in one programming language into an equivalent functional code in another programming language. There are three main types of computer language translators - *interpreters, compilers and assemblers.*

2.2.2 Application Software

Application software are computer programs that allow users to perform specific tasks. These programs are commonly referred to as *apps*, and are usually completely self-contained and commercially produced. They sit on top of systems software because they cannot run without the operating system and utilities. Application software can be used as a productivity/business tool; to assist with graphics and multimedia projects; to support home, personal, and educational activities; and to facilitate communications. Specific application software products, called software packages, are available from software vendors. Although application software are also available as shareware, freeware, and public-domain software, these usually have fewer capabilities than retail software packages. Examples include database programs such as MS-Access and Oracle, word processors like MS-word and word perfect, Spreadsheets like MS Excel, Lotus 1-2-3, Quattro Pro, and Open Office Calc, and Web browsers such as Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, and Apple Safari.

Review Questions

1. What are the major components of a computer system?
2. Give a schematic diagram of the functional components of a computer
3. Mention 5 input and output devices you know
4. What are the functions of the CPU
5. Write the full meaning of the following (i) EEPROM (ii) RAM (iii) ROM (iv) LED (v) LCD
6. What are the 2 types of viewing screens for monitors?
7. Differentiate between impact and non impact printers with examples
8. What is a computer software? Give two types of computer software
9. Give 6 examples of application packages you know
10. What are the functions of the following: (i) language translator (ii) utility program (iii) operating system (v) device drivers

Chapter Three

Number Systems

3.0 Introduction

Different forms of data and information such as audio, graphics, video, text, a numbers may all seem different to us. The computer however, stores all the internally in the same simple format: a sequence of 0's and 1's called *binary numbe* instead of the more familiar decimal system. A set of values used to represe different quantities is known as *Number System*. The computer number syste consists of the following:

- *Decimal Number System*
- *Binary Number system*
- *Octal Number system*
- *Hexadecimal Number system.*

The total number of digits used in a number system is called its *base*.

3.1 The Decimal Number System

The Decimal Number System is the most widely used number system. It consists ten digits (0,1,2,3,4,5,6,7,8,9). Thus, the base of decimal number system is 10. Thes digits can be used to represent any numeric value and the value represented b individual digit depends on weight and position of the digit.

Each number in this system consists of digits which are located at different position The position of first digit towards left side of the decimal point is 0. The position second digit towards left side of the decimal point is 1. Similarly, the position of fir digit towards right side of decimal point is -1. The position of second digit toward right side of decimal point is -2 and so on. For example, consider the number *497.86*

The position of **7** in the figure is **0**, the position of **9** is **1** and that of **4** is **2**. Similarly the position of **8** is **-1** and that of **6** is **-2**.

Again, the value of a decimal number is determined by multiplying the individu digits with the weight of their position and adding the results. The rightmost digit of number is the Least Significant Digit (LSD) and has the lowest weight while th

leftmost digit of a number is the Most Significant Digit (MSD) and has the highest weight. For example in the number 5423; the LSD is 3 and the MSD is 5.

Example 1: the weights and positions of each digit of the number 5423 are as follows:

position	3	2	1	0
weight	10^3	10^2	10^1	10^0
face value	5	4	2	3

The above table indicates that:

The value of digit 5 = 5×10^3 = 5000
The value of digit 4 = 4×10^2 = 400
The value of digit 2 = 2×10 = 20
The value of digit3 = 3×1 = 3

The actual value then is

$$5000 + 400 + 20 + 3 = 5423_{10}$$

Example 2: consider the number **139.78**; the weights and positions of each digit of the number **139.78** are as follows:

position	2	1	0	-1	-2
weight	10^2	10^1	10^0	10^{-1}	10^{-2}
face value	1	3	9	7	8

From the above table,

The value of digit 1 = 1×10^2 = 100
The value of digit 3 = 3×10^1 = 30
The value of digit 9 = 9×10^0 = 9
The value of digit 7 = 7×10^{-1} = 0.7
The value of digit 8 = 8×10^{-2} = 0.08

Adding them together, we obtain

$$100 + 30 + 9 + 0.7 + 0.8 = 139.78_{10}$$

- Digital Video Disk: Digital Video Disk (DVD) is similar to a CD but has larger storage capacity and enormous clarity. Depending upon the disk type, it can store several Gigabytes of data. DVDs are primarily used to store music or movies and can be played back on your television or the computer. They are typically not rewritable.

Solid state storage devices:

Unlike other storage devices discussed so far, solid-state storage devices have got no moving parts and data is stored and retrieved from them in a similar manner as it would be from computer memory. As a result, they are regarded as being robust and reliable. It is worthy to mention that this technology already exists in the form of flash memory used to store the Basic Input/Output System (BIOS) of a motherboard. Solid state storage devices come in different forms and sometimes under different commercial names such as **USB disk, Pen drives, Flash disks** etc. Some manufacturers tend to market these devices by adding components such as MP3 players within the devices and so on. However all of them work with very little power and represent a reliable device having relatively good storage capacity. Typical storage capacity ranges from 32Mb to 1GB.

C: Output Unit:

Output unit receives information from the CPU and presents it to the user in the desired form. The processed data, stored in the memory of the computer is sent to the output unit, which then converts it into a form that can be understood by the user. The output is usually produced in one of the two ways – on the display device, or on paper (hard copy). Some output devices are as follows:

- **Monitor**: Monitors, commonly called Visual Display Unit (VDU are the main output device of a computer. It forms images from tiny dots called pixels that

are arranged in a rectangular form. The sharpness of the image depends upon the number of pixels.

There are two kinds of viewing screen used for monitors.

- Cathode-Ray Tube (CRT)
- Flat-Panel Display

Cathode-Ray Tube (CRT) Monitor: the CRT display is made up of small picture elements called pixels. The smaller the pixels, the better the image clarity, or resolution. It takes more than one illuminated pixel to form whole character, such as the letter 'e' in the word help.

A finite number of characters can be displayed on a screen at once. The screen can be divided into a series of character boxes i.e. fixed locations on the screen where standard characters can be placed. Most screens are capable of displaying 80 characters of data horizontally and 25 lines vertically. Disadvantages of CRT include: large size and high power consumption

Flat-Panel Display Monitor: the flat-panel display refers to a class of video devices that have reduced volume, weight and power requirement in comparison to the CRT. You can hang them on walls or wear them on your wrists. Current uses of flat-panel displays include calculators, video games, monitors, laptop computer and graphics display. The flat-panel display is divided into two categories:

- Emissive Displays - The emissive displays are devices that convert electrical energy into light. Examples are plasma panel and LED (Light-Emitting Diodes).
- Non-Emissive Displays- They use optical effects to convert sunlight or light from some other source into graphics patterns. Example is LCD (Liquid-Crystal Device).

- **Printer**: Printers are used to produce paper (commonly known as hard copy) output. Based on the technology used, they can be classified as Impact or Non-impact printers.

I. *Impact Printers*:

These use the typewriting printing mechanism wherein a hammer strikes the paper through a ribbon in order to produce output. Characteristics of Impact Printers are:

- Very low consumable costs
- Very noisy

to a mass audience because they were smaller and relatively inexpensive than t predecessors. They proved to be highly reliable, faster and less human labou required. Computers developed then include IBM 360/370, PDP-8,11, CDC 6600.

Fourth Generation (1971-Present): the advent of microprocessor brought about fourth generation of computers, as thousands of integrated circuits (large sc integration and very large scale integration) were built onto a single silicon chip. T Intel 4004 chip, developed in 1971, located all the components of the computer from the central processing unit and memory to input/output controls—on a sin chip.

In 1981 IBM introduced its first computer for the home user, and in 1984 Ap introduced the Macintosh. Microprocessors also moved out of the realm of desk computers into many other areas of life as more and more everyday products began use microprocessors.

As these small computers became more powerful, they could be linked together form networks, which eventually led to the development of the Internet. Four generation computers also saw the development of GUIs (Graphical User Interfac the mouse and handheld devices.

Fifth Generation (Present and beyond): first to fourth generation computers did n have their own thinking power. Fifth generation computing devices, based o artificial intelligence are still in development, though there are some applications suc as voice recognition that are being used today. The use of parallel processing an superconductors are helping to make artificial intelligence a reality. The goal of fift generation computing is to develop devices that can respond to natural language inp and are capable of learning and self-organization. The programming languages LIS (List Processor) and PROLOG (Programming with Logic) are used for artifici intelligence.

It is believed that adopting quantum computatioh, molecular and nanotechnology i this era will radically change the face of computers in years to come.

Review Questions

- Define a computer.
- What are the 5 major tasks performed by the computer?
- What are the characteristics of a computer
- In what ways can the computer be classified?
- Give a brief history of computer
- Explain the following terms (i) Main frame (ii) micro computer (iii) hybrid computer (iv) digital computer (v) workstation
- What characteristics marked each generation of computers and give 2 examples of computers developed in each generation.
- Give the full meaning of the following: (i) PDA (ii) COBOL (iii) EDSAC (iv) FORTRAN (v) SSI (vi) GUI

PREFACE

Fundamentals of Computer Science & Basic Programming is *an eye opener* to the basics of computer Science and rudiments of basic programming. It is intended for beginners that wish to understand *where we were, where we are and where we are going* in the field of computer science in addition to exploring the act of programming.

The text is presented in a clear and concise format to help the reader digest the content easily with a lot of sample programs. It incorporates an overview of computer system, problem solving techniques, basic concepts of programming, features of VB.NET and an introduction to object-oriented programming.

With fundamentals of Computer Science & Basic Programming, you are in the right track to developing simple windows applications and console applications. This wonderful experience will definitely entice you to delve further into developing world class projects. *Remember the saying –a journey of thousand miles starts with a step.*

Finally, I must not fail to acknowledge all who in one way or the other made this piece of text possible; especially my darling husband and lovely children – *Kamsy* and *Ezeron*; for their prayers, encouragement and understanding. My heartfelt gratitude goes to my mum and siblings – Joy, Tom, Chekwube, Ralu and Charles, for their moral support. Suggestions and encouragements from my colleagues in the Department of Computer Science - Prof. Amigbogu, Dr. Ejiofor, Dr. Edebeatu, Dr. Umeh, Dr. Ekechukwu, Dr. Onyesolu, Dr. Obi, Dr. Okide, Dr. Mbeledogu, Dr. Ezeani, and Uncle Sam at one time or the other are also acknowledged. Inestimable thanks are also extended to other members of staff- Ositanwosu, Obinnu, Oluchukwu, Mrs Nwabueze, Dan and Sam. To the management and staff of De-Emeralds Printing and Publishing, I am very grateful for the role you played in the production of this text.

CONTENTS

- Useful for bulk printing due to low cost
- There is physical contact with the paper to produce an image

Character printers and Line printers fall under this category.

i. Character Printers: character printers are the printers which print one character at time. Examples include: Dot Matrix Printer (DMP) and Daisy Wheel printer.

- **Dot Matrix Printer:** in the market, one of the most popular printers is Do Matrix Printer. These printers are popular because of their ease of printing and economical price. Each character printed is in form of pattern of dots and head consists of a Matrix of Pins of size (5*7, 7*9, 9*7 or 9*9) which comes out to form a character; thus the name Dot *Matrix Printer.*

  Advantages
  - Inexpensive
  - Widely Used
  - Other language characters can be printed

  Disadvantages
  - Slow Speed
  - Poor Quality
- Daisy Wheel: The daisy wheel is a disk made of plastic or metal on which characters stand out in relief along the outer edge. To print a character, the printer rotates the disk until the desired letter is facing the paper. Then a hammer strikes the disk, forcing the character to hit an ink ribbon, leaving an impression of the character on the paper. These printers are generally used for word processing in offices which require a few letters to be sent here and there with very nice quality.

  Advantages
  - More reliable than Dot Matrix Printer (DMP)
  - Better quality
  - The fonts of character can be easily changed

  Disadvantages
  - Slower than DMP
  - Noisy

  - More expensive than DMP
  - Cannot print graphics

ii Line Printers: Line printers are the printers which print one line at a time. Examples of line printers are: drum Printer and chain Printer.

- **Drum Printer:** This printer is like a drum in shape so it is called drum printer. The surface of the drum is divided into number of tracks. One rotation of drum prints one line. Drum printers are fast in speed and can print 300 to 2000 lines per minute.

  Advantages
  - Very high speed

  Disadvantages
  - Very expensive
  - Characters fonts cannot be changed
- **Chain Printer:** In this printer, chains of character sets are used thus the name chain printer. A standard character set may have 48, 64, or 96 characters.

  Advantages
  - Character fonts can easily be changed.
  - Different languages can be used with the same printer.

  Disadvantages
  - Noisy

II. Non-impact Printers

Non-impact printers print the characters without using ribbon. They use chemical, heat or electrical signals to etch the symbols on paper. Non- impact printers print a complete page at a time and do not touch the paper while printing.

Characteristics of Non-impact Printers
- Faster than impact printers.
- They are not noisy.
- High quality.
- Support many fonts and different character sizes.

Chapter One

Computer: An Overview

What is a computer?

computer is an electronic device that accepts data inform of *input*, *processes* it ed on a program or sequence of instructions, *stores* it as desired and converts it to a meaningful state called *information* which is presented as output.

omputer therefore, basically performs five major tasks or functions irrespective of size and make. These are:

- Accepting data by way of input
- Storing the data
- Processing the data as required by the given instructions
- Giving result in form of output
- Directing the manner and sequence in which all of the above operations are performed.

he data or information could be inform of numbers, text, sound, image, animations nd video to mention only but a few.

.1 Characteristics of Computer

eatures of the computer that distinguish it from humans and other devices include:

- Speed: In general, no human being can compete with computer in solving complex computations faster. It takes only few seconds of the computer for calculations that could take man years to complete. Computers can perform millions

(1,000,000) of instructions and even more per second. Its speed is measured in Mega Hertz (MHz) or Giga Hertz (GHz).

- Accuracy: The degree of accuracy of computer is very high and every calculation is performed with the same accuracy. Since computer is programmed, its accuracy depends on instructions and data given to it. Wrong data and instructions will definitely yield inaccurate results. Thus, the old time slogan – *garbage in – garbage out* is very true.
- Storage: Computer has the ability to store a large amount of data. It has an in-built memory where it can store large volume of data and as well can support secondary storage devices. The storage capacity of a computer is measured in Bytes, Kilo Bytes (KB), Mega Byte (MB), Giga Byte (GB), Tetra Byte (TB).
- *Diligence*: Computer can work for hours without creating any error as opposed to humans that experience fatigue, lack of concentration, among others. Thus, it overpowers human being in routine work.
- Versatility: Computers can be applied in all areas of human endeavour such as medicine, sports, agriculture, banking, education and aviation. It can as well perform completely different tasks at the same time. One moment, it is genefating a student's transcript, the next moment it is busy making hotel reservations, and in between it may be helping an office secretary to trace an important letter in seconds.
- Power of Remembering: It can remember data for us. A computer can store and recall any amount of information because of its secondary storage capability. Every piece of information can be retained as long as desired by the user and it can be recalled at will instantaneously not minding how long it has been stored.
- No I.Q. (intelligent quotient). A computer is a mechanical device and has I.Q. of zero. It can only perform what it is programmed to do. Hence, only the user can determine what tasks a computer will perform.
- No feeling: Computer does not have emotions, knowledge, experience or feelings and thereby cannot be influenced.
- Reliability: Computer provides a very high speed accompanied by an equivalent level of reliability. Thus computers never make mistakes of their own accord.

you move the mouse, the pointer on the display screen moves in the sam direction.

- **Trackball**: A trackball is an input device used to enter motion data int computers or other electronic devices. It serves the same purpose as a mouse but is designed with a moveable ball on the top, which can be rolled in an direction.
- **Touchpad**: A touch pad is a device for pointing (controlling input positioning on a computer display screen. It is an alternative to the mouse. Originall incorporated in laptop computers, touch pads are also being made for use wit desktop computers. A touch pad works by sensing the user's finger movemen and downward pressure.
- **Touch Screen**: This allows the user to operate/make selections by simpl touching the display screen that is sensitive to the touch of a finger or stylus Widely used on ATM machines, retail point-of-sale terminals, car navigatio systems, medical monitors and industrial control panels.
- **Light Pen**: Light pen is an input device that utilizes a light-sensitive detector t select objects on a display screen.
- **Magnetic ink character recognition (MICR)**: MICR can identify characte printed with a special ink that contains particles of magnetic material. Thi device particularly finds applications in banking industry.
- **Optical mark recognition (OMR)**: Optical mark recognition, also called mar sense reader is a technology where an OMR device senses the presence o absence of a mark, such as pencil mark. OMR is widely used in tests such a aptitude test.
- **Bar code reader**: Bar-code readers are photoelectric scanners that read the ba codes or vertical zebra stripes marks, printed on product containers. These devices are generally used in super markets and bookshops.
- **Scanner**: Scanner is an input device that can read text or illustrations printed on paper and translates the information into a form that the computer can use. A scanner works by digitizing an image.

B: The Central Processing Unit:

Central Processing Unit (CPU) is usually referred to as the brain of the computer. It is the computing center of the system and consists of the following features:

- CPU performs all types of data processing operations.
- It stores data, intermediate results and instructions (program).
- It controls the operation of all parts of computer.

The CPU consists of three components: the control unit, an arithmetic & logic unit, and internal storage unit.

- **The Control Unit** — the control unit maintains order within the computer and directs the flow of traffic (operations) and data. It does not perform actual processing operations on the data; instead, it selects one program statement at a time from the program storage area, interprets the statement, and sends the appropriate electronic impulses to the appropriate unit for necessary actions. Again, it communicates with Input / Output devices for transfer of data or results from storage.
- **Arithmetic and Logic Unit ((ALU)**: all calculations and comparisons, based on the instructions provided, are carried out within the ALU. It performs arithmetic functions like addition, subtraction, multiplication, division and also logical operations like greater than, less than and equal to.
- **Memory Unit:** the function of the memory unit is to store programs and data. It stores instructions, data and intermediate results. Information is supplied by this unit to other units of the computer when needed. It is also known as internal storage unit, main memory, primary memory or Random access memory (RAM). Basically, functions of memory unit are:
  - It stores all the data and the instructions required for processing.
  - It stores intermediate results of processing.
  - It stores final results of processing before these results are released to an output device.
  - All inputs and outputs are transmitted through main memory.

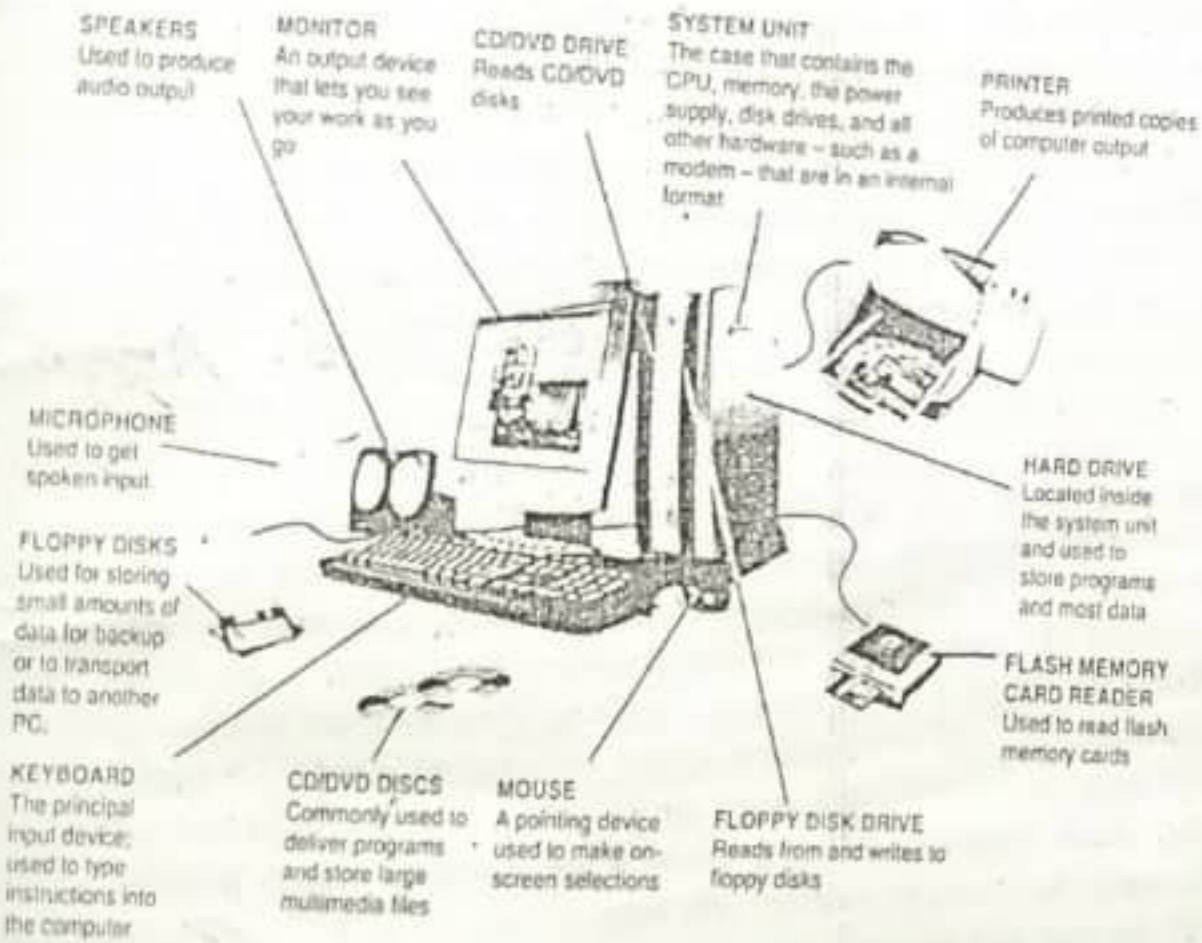
The size of the memory affects speed, power and capability of the computer. Primary memory and secondary memory are two types of memories in the computer.

Chapter Two

Basic Components of a Computer System

2.0 Introduction

The computer system is a group of integrated parts that have the common purpose of performing various operations. It consists of a computer, all the support equipment necessary for its use and instructions that specify the performance of certain tasks.



A computer system consists of two major elements: **hardware** and **software.** Computer hardware is the collection of all the physical parts (as shown in figure 2.1). Computer software, on the other hand, is not something you can touch; it is a set of instructions for a computer to follow and perform specific operations. Both hardware and software are required for a computer system to work.

2.1 Functional Components of a Digital Computer

In order to carry out its basic operations, the computer is made up of three distinct units. These are the input unit, the central processing unit (CPU) and the output unit as depicted in figure 2.2 below.

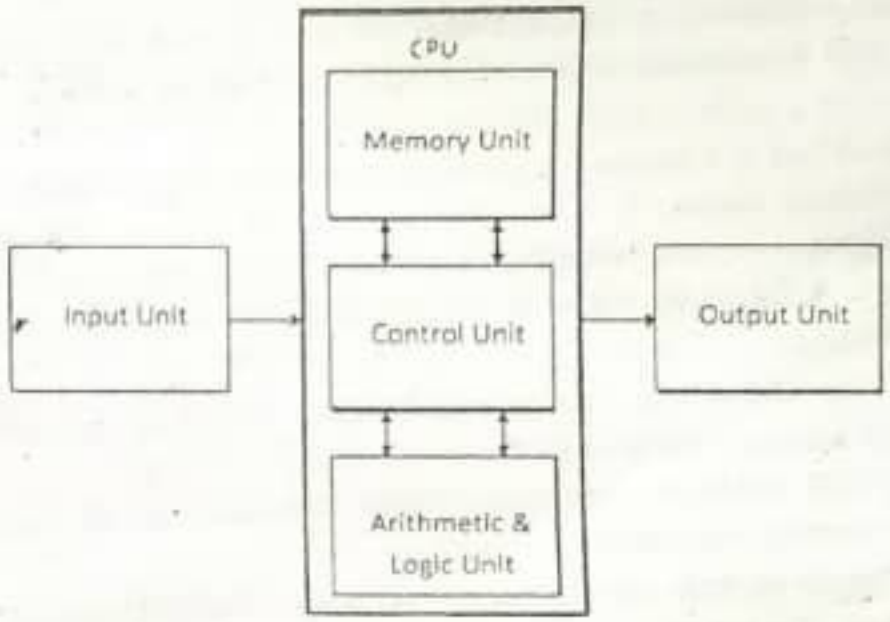


Figure 2.2: Functional components of a computer

A: The Input Unit:

This unit is used for entering data and programs into the computer. Data are read from a source, such as magnetic disks, and translated into electronic impulses for transfer to the CPU. Some typical input devices are as follows:

- **Keyboard**: The keyboard is very much like a standard typewriter keyboard with a few additional keys. The basic QWERTY layout of characters is maintained to make it easy to use the system. The additional keys are included to perform certain special functions. These are known as function keys that vary in number from keyboard to keyboard.
- **Mouse**: A device that controls the movement of the cursor or pointer on a display screen. A mouse is a small object you can roll along a hard and flat surface. Its name is derived from its shape, which looks a bit like a mouse. As

instructions that control it are built directly into the computer, which makes for a more efficient and effective operation. Such a computer system can be used in traffic lights control, navigational system in an aircraft, weather forecasting, satellite launch / tracking, oil exploration, and in ATM.A drawback of this specialization, however, is the computer's lack of versatility.

1.3.1: According to type of data:

Considering the type of data being handled, computers are grouped into analog, digital or hybrid.

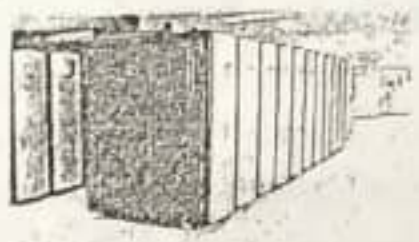
- *Analog computers* are used to process continuous data. They represent variables by physical quantities such as temperature, pressure, angular position or voltage. These quantities are continuous and have an infinite variety of values. Thus, their results are approximately correct. Examples include the speedometer of a car, voltmeter, thermometer and trye pressure gauge.
- *Digital computers* recognize data by counting discrete signal representing either a high or low voltage state of electricity. They process data into a digital value (in 0s and 1s) and give the results with more accuracy and at a faster rate.
- *Hybrid computers* incorporate the measuring feature of an analog computer and controlling feature of a digital computer. It offers a cost effective method of performing complex simulations. Hybrid computers are mostly used in hospitals such as the ECG, industries, aviation and scientific research institutes.

1.3.2 On the basis of Size

Based on size and functionality, computers can be classified as supercomputers, mainframe computers, minicomputers and microcomputers.

Super Computers

These are the largest, fastest, most powerful and most expensive type of computers. They are employed for specialized applications that require immense amounts of mathematical calculations such as weather forecasting, animated graphics and fluid dynamic. Other applications include processing of geological data, nuclear energy research, and petroleum exploration. Examples of supercomputers are: CRAY 3, NEC-500, PARAM9000, PARAM1000 and ANURAG.

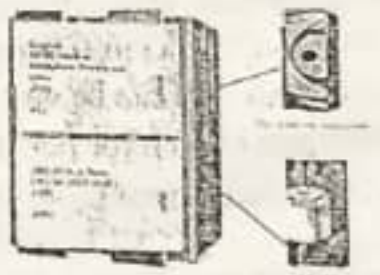


CRAY xt3 7242: (Super Computer)

Supercomputers can perform up to 128 gigaflops and use bus widths of 32 or 64 bits. This capability makes supercomputers suitable for processor-intensive applications, such as graphics. The speed of modern supercomputers is measured in **nanoseconds** and **gigaflop**. A nanosecond is one billionth of a second and a gigaflop is one billion floating-point arithmetic operations per second.

Mainframe Computers

Mainframe computers are very large and expensive computers capable of supporting hundreds, or even thousands, of users simultaneously. One often fills an entire room. Terminals can be located in the same room with the mainframe computer or in different rooms, buildings, or cities. Large businesses, government agencies, and universities usually use this type of computer.



The IBM Smart Analytics System 9700 (mainframe computer)

In some ways, mainframes are more powerful than supercomputers because they support more simultaneous programs. But supercomputers can execute a single program faster than a mainframe. In other words, a supercomputer channels all its power into executing a few programs as fast as possible, whereas a mainframe uses its power to execute many programs concurrently. Examples of mainframe computers are CDC 600, IBM S/390, Amdahl 580, Control Data Cyber 176, VAX 8842, IBM 3090/600 and IBM 4381.

Inkjet, DeskJet, Laser, Thermal printers fall under this category of printers. Le look at the features of laser printer and inkjet printer.

i. **Laser Printers:** These printers use laser lights to produce the dots needed form the characters to be printed on a page.

Advantages

- Very high speed
- Very high quality output
- Give good graphics quality
- Support many fonts and different character sizes

Disadvantages

- Expensive.
- Cannot be used to produce multiple copies of a document in single printing.

ii. **Inkjet Printers:** Inkjet printers are non-impact character printers based on relatively new technology. They print characters by spraying small drops ink onto paper. Inkjet printers produce high quality output with presentab features. They make less noise because no hammering is done and have man styles of printing modes available. Colour printing is also possible and som models of Inkjet printers can as well produce multiple copies of printing.

Advantages

- High quality printing
- More reliable

Disadvantages

- Expensive as cost per page is high
- Slow as compared to laser printer

❖ **Plotter:** Plotters are used to print graphical output on paper. It interprets computer commands and makes line drawings on paper using multi-colour automated pens. It is capable of producing graphs, drawings, charts and maps.

❖ **Facsimile (FAX):** Facsimile machine, a device that can send or receive pictures and text over a telephone line. Fax machines work by digitizing an image.

❖ **Sound cards and Speaker(s):** An expansion board that enables a computer to manipulate and output sounds. Sound cards are necessary for nearly all CD-ROMs and have become commonplace on modern personal computers. Sound cards enable the computer to output sound through speakers connected to the board, record sound input from a microphone connected to the computer, and manipulate sound stored on a disk.

2.2 Computer software

Computer software, or simply *software* is any set of machine-readable instructions that directs a computer's processor to perform specific operations. For example, without your Internet browser, you cannot surf the Internet and without an operating system, the browser cannot run on your computer. Computer software is non-tangible, contrasted with computer hardware, which are the physical components of the computers. Computer software is broadly classified into system software and application software.

2.2.1 System software

This is computer software designed to operate and control the computer hardware and to provide a platform for running application software. System software can be separated into *operating systems, utilities, device drivers and language translators.*

Operating system: this performs the following functions:

- Manages computer resources: this it does by
  - Coordinating memory, processing storage, and printers
  - Monitoring system performance
  - Providing security
  - Starting-up the computer
- Provides user interface such as Character-based interface and Graphical User Interface (GUI)
- Runs applications: these include multitasking, foreground & background applications

Examples of operating systems are Windows, UNIX, LINUX and Mac OS

Utilities: these are specialized programs that make computing easier. They help to perform maintenance or correct problems with a computer system. Utility programs

- Automation: Once the instructions are fed into the computer it works automatically without any human intervention.

1.2 History of computers

Computers have infiltrated every aspect of our society ranging from simple addition and subtraction to more complex robot manipulations. Most of our day to day activities cannot be accomplished without using computers. To fully understand and appreciate the impact computers have on our lives and promises they hold for the future, it is important to understand their evolution.

What is known as computer today dates back to the era of Abacus in 300BC. The *abacus* was an early aid for mathematical computations. Its only value was in assisting the memory of the human performing the calculation. A skilled abacus operator could work on addition and subtraction problems at the speed of a person equipped with a hand calculator though multiplication and division were slower.

In 164_ Blaise Pascal, at age 19, invented the *Pascaline* as an aid for his father who was a tax collector. However, the Pascaline was limited to only addition. A German mathematician and philosopher, Gottfried Wilhem von Leibniz, in 1674, improved the Pascaline by creating a machine that could multiply as well as perform addition. In 1801 the Frenchman Joseph Marie Jacquard invented a power loom that could base its weave upon a pattern automatically read from punched wooden cards held together in a long row by rope. By 1822 the English mathematician *Charles Babbage* was proposing a steam driven calculating machine; the size of a room, which he called the *Difference Engine*. This machine would be able to compute tables of numbers, such as logarithm tables. He obtained government funding for this project due to the importance of numeric tables in ocean navigation. But construction of Babbage's Difference Engine proved exceedingly difficult and ten years later the device was still nowhere near complete, acrimony abounded between all involved, and funding dried up. The device was never finished. Babbage was not deterred, and by then was on to his next brainstorm, which he called the *Analytic Engine*. This device, large as a house and powered by 6 steam engines, would be more general purpose in nature because it would be programmable with the punched card technology of Jacquard. However, the Analytic Engine remained unbuilt but gave rise to the concept of

subroutine and looping used in today's programming. In 1896, Herman Hollerith built a company, the Tabulating Machine Company which, after a few buyouts, eventually became International Business Machines, known today as *IBM*. In 1938, William Hewlett and David Packard co-found Hewlett-Packard (HP). John V. Atanasoff, a professor at Iowa State College and his graduate student, Clifford Berry, envisioned an all-electronic computer that applied Boolean algebra to computer circuitry. This approach was based on the mid-19th century work of George Boole (1815-1864) who clarified the binary system of algebra, which stated that any mathematical equations could be stated simply as either true or false. By extending this concept to electronic circuits in the form of on or off, Atanasoff and Berry developed the first all-electronic computer by 1940 using radio valves as switches. Their project, however, lost its funding and their work was overshadowed by similar developments by other scientists.

Howard Aiken, in 1944 built the world's first programmable digital computer made in the U.S. It was called the *Mark I* or *ASCC (Automatic Sequence Controlled Calculator)*. *Mark I* was built as a partnership between Harvard and IBM. But it was not a purely electronic computer but rather an electromechanical machine. The Mark I was constructed out of switches, relays, rotating shafts, and clutches. Programs and information were input by a punched paper ribbon. Aiken had of course managed to make the "analytic engine" that Babbage had envisioned. Unfortunately, the machine was very slow, taking seconds to multiply or divide six digit numbers. One of the giants in the development of high level languages on digital computers was Grace Murray Hopper, recruited by Aiken in 1943 to be the third programmer of the Mark I. Starting in 1959, she led the design team for the COBOL (Common Business Oriented Language) in association with Sperry Univac. Hopper was credited with coining the term *bug* in reference to a glitch in the machinery. The first bug was actually a moth which flew through an open window and into one of the computer relays of the Mark I, temporarily shutting down the system. The moth was removed and pasted into the logbook. From then on, *debugging* turns out to be the detection and correction of program errors.

By replacing the relays with electronic switches (vacuum tubes) which rely on the movement of electrons rather than the slow physical movement of mechanical

The primary memory: this can be further classified as Random Access Memory (RAM) and Read Only Memory (ROM).

RAM: this is the unit in a computer system. It is the place in a computer where the operating system, application programs and the data in current use are kept temporarily so that they can be accessed by the computer's processor. It is said to be *volatile* since its contents are accessible only as long as the computer is on. The contents of RAM are no more available once the computer is turned off. RAM is of two types: Static RAM (SRAM) and Dynamic RAM (DRAM).

- Static RAM (SRAM): The word static indicates that the memory retains its contents as long as power is being supplied. SRAM need not have to be refreshed on a regular basis. It uses more chips than DRAM for the same amount of storage space, thus making the manufacturing costs higher. SRAM is used as cache memory and has very fast access.
- Dynamic RAM (DRAM): DRAM, unlike SRAM, must be continually refreshed in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory because it is cheap and small.

ROM: this is a special type of memory which can only be read. Its contents are not lost even when the computer is switched off. It typically contains manufacturer's instructions among other things. ROM also stores an initial program called the 'bootstrap loader' whose function is to start the operation of computer system once the power is turned on. There are different types of ROM and these are discussed below.

Types of ROM:

- Programmable Read-Only Memory (PROM): This type of ROM can be re-programmed by using a special device called a PROM programmer. Generally, a PROM can only be changed / updated once.
- Erasable Programmable Read-Only Memory (EPROM): This type of ROM can have its contents erased by ultraviolet light and then reprogrammed by a RPROM programmer. This procedure can be carried out many times;

however, the constant erasing and rewriting will eventually render the chip useless.

- Electrically Erasable Programmable Read-Only Memory (EEPROM): This type of ROM works in a similar way as *flash memory* in that its contents can be *flashed* for erasure and then written to without having to remove the chip from its environment. EEPROMs are used to store a computer system's BIOS, and can be updated without returning the unit to the factory. In many cases, BIOS updates can be carried out by computer users wishing a BIOS update.

Note:

i. Flash memory is an example of quite a recent type of storage technology known as **solid state devices**. This type of portable storage has become very popular because of its low price and high storage capacity compared to its rivals, such as the floppy disk.

ii. Unlike ROM, flash memory can be read from and written to and unlike RAM does not require power to retain its data.

Secondary Memory: RAM is a volatile memory with limited storage capacity. Secondary/auxiliary memory on the other hand, is a non- volatile storage; data written to it are not loss when the computer goes off. The data stored in it is permanent but can be deleted if desired. However, the CPU does not directly access these memories; instead they are accessed via input-output routines. Contents of secondary memories are first transferred to main memory and then CPU accesses it. Examples of secondary storage are:

- Hard Disk: hard disks are made up of rigid materials and are usually a stack of metal disks sealed in a box. The hard disk and the hard disk drive exist together as a unit and is a permanent part of the computer where data and programs are saved. These disks have storage capacities ranging from 1GB to 80 GB and more. Hard disks are rewritable.
- Compact Disk: compact Disk (CD) is a portable disk having data storage capacity between 650-700 MB. It can hold large amount of information such as music, full-motion videos and text. CDs can be either read only or read / write type.

3.2 The Binary Number System

As stated earlier, digital computer represents all kinds of data and information in th binary number system. Binary Number System consists of two digits [0, 1]; showin that each digit in binary can either be a '0' or a '1'. The base is therefore 2.

When we count up from zero in binary, we run out of symbols much more frequentl i.e. 0, 1,(.. ooop'), there are no more symbols. We do not go to 2 because in binary a 2 does not exist. Instead, we use 10, meaning 2 in decimal.

A combination of binary numbers may be used to represent different quantities lik 11011. The weight of each position is a power of 2. For example: the place value o the digits of the number **11011** according to position and weight is as follows:

Face value	1	1	0	1	1
Position	4	3	2	1	0
Weight	2^4	2^3	2^2	2^1	2^0

The binary system is useful in computer science and electrical engineering. It is used by transistors found in practically all electronic devices. A '0' means no current, and a '1' means to allow current. With various transistors turning on and off, signals and electricity is sent to do various things such as making a call or printing these letters on paper.

3.3 The Octal Number System:

Octal Number System consists of eight digits [0, 1, 2, 3, 4, 5, 6, 7]. The maximum digit in octal is seven (7). Thus the number 3489 is not valid since 8 and 9 are not valid digits in octal. The base of octal system is 8 and each digit position in this system represents a power of 8. Consider the number 4573; the place value, position and weight of the digits are as shown:

Face value	4	5	7	3
Position	3	2	1	0
Weight	8^3	8^2	8^1	8^0

By multiplying the place (face) value by the corresponding weight, we obtain the decimal equivalent of the number. Thus 4573 in octal is equivalent to

$$4x\ 8^3+5x8^2+7x8^1+3x8^0$$
$$= 4x512+5x64+7x8+3x1$$
$$= 2048 + 320 + 56 + 3$$
$$= 2427_{10}$$

Octal number system can be used to represent long binary numbers.

3.4: The Hexadecimal Number System

The Hexadecimal Number System consists of 16 digits : 0 to 9 and A to F; [0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F]; where the letters A to F represent decimal numbers from 10 to 15. The base of this number system is 16 and each digit position in hexadecimal system represents a power of 16. This number system can also be used to represent long binary numbers. Consider the number **87AB**; the place value of each digit according to position and weight is as follows:

Face value	8	7	A	B
Position	3	2	1	0
Weight	16^3	16^2	16^1	16^0

By multiplying the place value by the corresponding weight, we obtain the decimal equivalent of the number. Thus 87AB in hexadecimal is equivalent to

$$8x\ 16^3+7x16^2+Ax16^1+Bx16^0$$
$$= 8x4096+7x256+10x16+11x1$$
$$= 32768 + 1792+160+11$$
$$= 34731_{10}$$

3.5 Conversion from base 10 to other bases

In order to convert a decimal number into its representation in a different number base, we repeatedly divide the decimal number by the base in which it is to be converted to, until the quotient becomes zero. As the number is divided, the remainders in reverse order form the digits of the number in the converted base.

Example 3.4.1: Convert 54_{10} to binary (base 2), octal (base 8) and hexadecimal (base 16)

Solution:

Review Questions

1. What is LSD and MSD
2. What are the common number systems used by the computer
3. Convert the following numbers in decimal to (i) binary (ii) octal (iii) hexadecimal
   i. 28 ii. 33.14 iii. 56.325 iv. 79
4. Convert the following numbers to decimal
   i. $(111.1101)_2$ ii. $(100110011)_2$ iii. $(764.27)_8$ iv. $(346)_8$
   v. $(AD2B.19)_{16}$ vi. $(167D)_{16}$
5. Convert the following binary numbers to (i) octal (ii) hexadecimal by (a) direct conversion (b) by converting first to decimal and then to the desired base
   i. 1101111 (ii) 01011111.11 (iii) 101010101 (iv) 1011110011

Chapter four

Arithmetic Operations

4.1 Introduction.

Just as we can perform addition, subtraction, division and multiplication in decimal, we can equally do so in other bases.

Example 4.1.1: perform the following arithmetic operations in octal
(i) 234 + 567 (ii) 65 + 3765 (iii) 653 – 476 (iv) 57 x 46

Solution:

i. 234 + 567

```
  274
+
  567
 1063
```

Explanation:
We start by adding 4 and 7 = 11_{10}; dividing it by 8 (since we are working in base 8; we should visualize 8 as our conventional decimal) will give us 1 R 3; we then write 3 and carry 1 to the next digit (7); this will give us 8. We then add 8 and 6 to produce 14_{10}; again we divide by 8 to obtain 1 R 6. This 6 is written down and we carry 1 to the next digit (2) to produce 3 which we then add to 5 to yield 8_{10}. We again divide by 8; giving us 1 R 0. We put down 0 and carry 1 to next digit (invisible 0). Finally, we are done.

ii. 3765 + 65

```
  3765
+
    65
 4052
```

The same process as in i above was followed.

iii. $1011.001_2 - 110.10_2$

Step 1: we add 0's to 110.10 to make the number of digits equal i.e.
110.10 = 0110.100

Step 2: we find the 1's complement of the subtrahend
1's complement of 0110.100 = 1001.011

Step 3: add to the minuend

1011.001
\+ 1001.011
10100.100

Step 4: There is an extra digit (end carry); so we add 1 to the last bit

10100.100
\+ 1
10100.101

Thus, $1011.001_2 - 110.10_2 = 10100.101_2$

iv. $10110.01_2 - 11010.10_2$

Solution:

Step 1: both have equal no of digits, so we move to step 2

Step 2: we find the 1's complement of the subtrahend
1's complement of 11010.10 = 00101.01

Step 3: add to the minuend

10110.01
\+ 00101.01
11011.10

Step 4: There is no extra digit (end carry) showing that the result is negative; so we jump to step 5

Step 5: find the 1's complement of the result
11011.10 = 00100.01

$10.01_2 - 11010.10_2$ is therefore - $00100.01_2 = -100.01_2$

Binary subtraction using 2's complement

perform binary subtraction using 2's complement, the following steps are taken:

i. Ensure that the number of digits in both the minuend and subtrahend are equal
ii. Find the 2's complement of the subtrahend
iii. Add it to the minuend
iv. If there is an extra digit (end carry), this is discarded and the result is positive
v. If there is no extra digit, the result is negative and we find the 2's complement of the result

xample 4.4.1: perform the following binary subtractions using 2's complement
$110110_2 - 10110_2$ (ii) $1011_2 - 111001_2$ (iii) $1010.11_2 - 1001.01_2$

i. 110110_2 - 10110_2

lution

Step 1: we make the number of digits equal
10110 = 010110

Step 2: we find the 2's complement of the subtrahend
1's complement of 010110 = 101001
2's complement = 101001 + 1 = 101010

Step 3: add to the minuend

110110
\+ 101010
1100000

Step 4: There is an extra digit (end carry) showing that the result is positive; we therefore discard it

Thus $110110_2 - 10110_2 = 100000_2$

ii. $1011_2 - 111001_2$

Step 1: we make the number of digits equal
1011 = 001011

Step 2: we find the 2's complement of the subtrahend
1's complement of 111001 = 000110
2's complement = 000110 + 1 = 000111

Step 3: add to the minuend

```
  001011
+ 000111
  010010
```

Step 4: There is no extra digit (end carry) showing that the result is negative; we therefore jump to step 5
Step 5: find the 2's complement of the result

1's complement of 010010 = 101101
2's complement = 101101 + 1 = 101110

Thus $1011_2 - 111001_2 = -101110_2$. Same result obtained when we used the 1's complement.

iii. $1010.11_2 - 1001.01_2$

Step 1: number of digits are equal so we proceed to step2

Step 2: we find the 2's complement of the subtrahend
1's complement of 1001.01 = 0110.10
2's complement = 0110.10 + 1 = 0110.11

Step 3: add to the minuend

```
  1010.11
+ 0110.11
  10001.10
```

Step 4: There is an extra digit (end carry) showing that the result positive; we therefore discard it

Thus, $1010.11_2 - 1001.01_2 = 0001.10_2 = 1.10_2$

Review Questions

Perform the following arithmetic operations as indicated:

1. In binary
   i. (11001) + (1110)
   ii. (1111.11) + (1011.10)
   iii. (11) x 101
   iv. 2's complement of 1110111
   v. 1's complement of 01010101
   vi. Subtract 11011 from 111111 using 1's complement and 2's complement
   vii. Subtract 11011011 from 1011 using both 1's and 2's complement
2. In octal
   i. Subtract 124 from 246
   ii. 12 x 14
   iii. 764 + 1262
   iv. 13.2 x 5
3. In hexadecimal
   i. ABF + DDE
   ii. AB X 3
   iii. F201 – AA
   iv. 13.6 X 9

iii. 653 – 476

```
  653
-
  476
  155
```

Note: the subtraction of octal numbers follows the same rules as the subtraction of numbers in any other number system. The only variation is in borrowed number. In the decimal system, you borrow a group of 10_{10}. In the octal system you borrow a group of 8_{10}.

Explanation of the result:

3 – 6 cannot go, we then borrow 1 (i.e a group of 8_{10}) from the next digit (5-1) to give us 11 (3 + 8); we then subtract 6 from 11 to obtain 5. The next digit has reduced to 4, again, 4 -7 cannot go so we borrow 1 (group of 8_{10}) from the next digit on the left (6 - 1) to give us 12 (4 +8); we then subtract 7 from it to obtain 5. Again, the 6 on the left has reduced to 5; we then subtract 4 from it to obtain 1. So easy hmm....

iv. 57 x 46

```
   57
 x
   46
  432
 274
 3372
```

Explanation:

7 x 6 = 42; we divide by 8 to obtain 5 R 2; we put down 2 carry 5, 6x5 = 30, we then add the 5 we carried to obtain 35. Divide again by 8 to get 4 R 3 , put down 3 and carry 4 to the next digit. The process continues ... .

Example 4.1.2: perform the following arithmetic operations in base 16 (i) FE + 2A (ii) 34C8 + 911

i.

```
  FE
+
  2A
 128
```

ii.

```
  34C8
+
   911
  3DD9
-
```

Example 4.1.3: perform the following binary arithmetic (i) 110 + 101 (ii) 011101 + 11101 (iii) 1011 x 11

```
   110
+
   101
  1011
```

Note: the maximum digit in base 2 is 1; therefore, 1+1 will produce a zero carry 1.

ii.

```
  1011101
+
   11101
 1111010
```

iii.

```
   1011
 x
     11
   1011
  1011
 100001
```

4.2 Binary system complements

Computers perform most arithmetic operations using addition operation. This makes circuit design simple and with only one circuit you can do different things. Complements are used in the digital computers to simplify the subtraction operation using addition. Binary system has two types of complements: 2's complement and 1's complement.

2's complement is especially important because it allows us to represent signed numbers in binary, and 1's complement is the interim step to finding the 2's complement.

Chapter five

Boolean Algebra

5.0 Introduction

Boolean algebra, first developed by 19th century mathematician George Boole, is mathematical formalization of intuitive logic. It provides a formal framework f reasoning about relationships between mathematical quantities that are limited to tw possible values: **true** and **false**, often denoted as 1 and 0 respectively. In particul Boolean algebra allows us to reason about *operations* on truth values. They take o or more truth values as input, and output a single truth value.

This is greatly applied in the design of logic gates which are the basic building bloc of any digital system. Logic gate is an electronic circuit having one or more than o input and only one output.

To help show the function of a logic gate, truth tables are used. They show how t input(s) of a logic gate relate to its output(s). The gate input(s) are shown in the le column(s) of the table with all the different possible input combinations and t output(s) are shown in the right hand side column. For any n inputs, there are possible input combinations. For example, if there are 2 inputs then there will be possible combinations, and if there 3 inputs then there will be 8 possible inp combinations.

5.1 Logic Gates

There are eight basic logic gates: OR, AND, NOR, NAND, NOT, X-OR, X-NOR a buffer gates.

The OR gate

The output from an OR gate is true if any of the inputs is true otherwise it is fals This means that the output is 1 except when all inputs are 0. It uses the operator (read as or). The table below shows the symbol and truth table for the OR gate.

Logic gate	Symbol	Truth table		
OR	A, B, Y	A	B	Y= A + B
		0	0	0
		0	1	1
		1	0	1
		1	1	1

The AND gate

he output is 1 when all the inputs are 1 otherwise it is 0. This shows that even when ll inputs less one is true; the output will still be false. It uses the operator (.) just like he multiplication symbol. Below is the symbol and truth table for AND gate.

Logic gate	Symbol	Truth table		
AND	A, B, Y	A	B	Y= A.B
		0	0	0
		0	1	0
		1	0	0
		1	1	1

The NOT (inverter) gate

This gate is applicable to only one input. It produces the inverse of the input; if the input is 0, the output is 1 and if the input is 1, the output is 0. The inverter gate uses bar (----) operator.

Logic gate	Symbol	Truthtable	
NOT	A, Y	A	B =$\bar{A}$
		0	1
		1	0

The Buffer gate

This gate is again applicable to only one input. A buffer has only a single input and a single output with behavior that is the opposite of a NOT gate. It simply passes its

1's complement

The 1's complement of a number is found by simply changing all 1's to 0's and all 0's to 1's in the given number. For example, find the 1's complement of 10101 :

Given number	1	0	1	0	1
	↓	↓	↓	↓	↓
1's complement	0	1	0	1	0

2's complement

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

i.e. 2's complement = 1's complement + 1. For example, find the 2's Complement of 10101

Given number	1	0	1	0	1
	↓	↓	↓	↓	↓
1's complement	0	1	0	1	0
Add 1					1
Result	0	1	0	1	1

4.3 Binary subtraction using 1's complement

Given two binary numbers A and B; and we are required to subtract B from A using the 1's complement method; the following steps are taken:

i. Make sure both A and B has equal number of digits by appending zero(s) if necessary
ii. Write down 1's complement of the subtrahend (i.e. B).
iii. Add this to the minuend(i.e. A)
iv. If the result of the addition has end carry, bring it down and add it to the last bit.
v. If there is no end carry, the result is negative and we again obtain 1's complement of the result to get the final result.

Arithmetic Operations

Example 4.3.1 : evaluate (i) $(110101)_2 - (100101)_2$ (ii) $1011_2 - 111001_2$ (iii) $011.001_2 - 110.10_2$ (iv) $10110.01_2 - 11010.10_2$

Solution: (i)

Step 1: both has 6 digits each, so we proceed to step 2

Step 2: we find the 1's complement of the subtrahend
1's complement of 100101 = 011010

Step 3: add to the minuend

```
  110101
+ 011010
--------
 1001111
```

Step 4: There is an extra digit (end carry); so we add 1 to the last bit

```
  001111
+      1
--------
  10000
```

Thus, $110101_2 - 100101_2$ is 1000_2

ii. $1011_2 - 111001_2$

Solution

Step 1: we add 0's to 1011 to make the number of digits equal i.e.
1011 = 001011

Step 2: we find the 1's complement of the subtrahend
1's complement of 111001 = 000110

Step 3: add to the minuend

```
  001011
+ 000110
--------
  010001
```

Step 4: There is no extra digit (endcarry); so we jump to step 5

Step 5: find the 1's complement of the result
010001 = 101110

Hence, $1011_2 - 111001_2 = -101110_2$

input, unchanged, to its output. In a boolean logic simulator, a buffer is mainly us
to increase propagation delay. In a real-world circuit, a buffer can be used to ampli
a signal if its current is too weak; it can increase the capability of the output of o
gate to drive a
number of other gates.

Logic gate	Symbol	Truth table	
		A	Y = A
Buffer	A — Y	0	0
		1	1

The NAND (Not AND) gate

Output from a NAND gate is the inverse of AND gate. It is a 1 for all combinations of inputs except when all the inputs are 1. Below is an illustration of the symbol and truth table.

Logic gate	Symbol	A	B	$Y = \overline{A.B}$
NAND	A, B → Y	0	0	1
		0	1	1
		1	0	1
		1	1	0

NOR (Not OR) gate

The output is the inverse of the result of OR gate. It is a 1 if and only if all inputs ar
0 otherwise it is a 0.

Logic gate	Symbol	A	B	$Y = \overline{A+B}$
NOR	A, B → Y	0	0	1
		0	1	0
		1	0	0
		1	1	0

xclusive - OR (XOR) gate

OR is a variation of the OR gate. Here, the output is 1 when any of the inputs is 1
t not when all is 1. That means that for a 2-input XOR gate, the output is 1 for (A
d NOT B) or (B and NOT A).

Logic gate	Symbol	Truth table		
		A	B	$Y=A \oplus B$ or $(\overline{A}.B) + (\overline{B}.A)$
XOR	A, B → Y	0	0	0
		0	1	1
		1	0	1
		1	1	0

xclusive - NOR (XNOR) gate

his is the inverse of the XOR gate. The output is 1 when both inputs are the same
d 0 when they have different values. The symbol and truth table is as shown

Logic gate	Symbol	A	B	$Y = \overline{A \oplus B}$ $(\overline{A}.\overline{B} + A.B)$
XNOR	A, B → Y	0	0	1
		0	1	0
		1	0	0
		1	1	1

4. What is the only set of input conditions that will produce a LOW output for any OR gate?
5. What are the basic logic gates?
6. Simplify the following
   i. $Y = A\bar{B}D + A\overline{B.D}$
   ii. $Z = (\bar{A} + B)(A + B)$
   iii. $Y = A\bar{C} + AB\bar{C}$
   iv. $\overline{AB}C\bar{D} + ABCD$
7. What is Boolean algebra and truth table
8. True or false: An AND gate output will always differ from an OR gate output for the same input conditions?
9. a. How do you interpret (order of precedence) the expression: A•B+C ? – Is it
   i. A •B ORed with C ?
   ii. A ANDed with B+C ?
   b. draw the logic circuit of the expression

Chapter Six

Program Development

1 Overview of programming languages

Computers as we all know cannot function without a computer program. A *computer program* however, is the set of instructions given to the computer to perform a given task. These set of instructions are written using a computer programming language which is the language with which we can communicate with the computer. Programming languages are broadly classified into low level and high level languages. Low level languages are closer to the machine and examples are machine language and assembly language. High level languages on the other hand are closer to human understanding. Examples are C++, visual C#, Java and Visual Basic.NET.

For effective communication, every program written in either assembly language or high level language needs to be translated to its machine equivalent (sequence of 0's and 1's) which is the only language the computer understands. A translator called an *assembler* is required to translate an assembly language to the corresponding machine language while a compiler or interpreter is required to translate a high level language to the corresponding machine language. The section below describes the features of the different levels of programming languages.

1.1 Machine Language

This is a programming language that is interpreted and executed directly by the computer. Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed. It has its merits and demerits as depicted in the table below

Advantages	Disadvantages
• Machine language makes fast and efficient use of the computer.	• All operation codes have to be remembered • Very difficult to write as only strings of 0's and 1's is used for every instruction • Machine dependent.
• It requires no translator to	• All memory addresses have to be

5.2 Laws of Boolean algebra

Boolean algebra is a system of mathematics based on logic that has its own set of rules or laws which are used to define and reduce Boolean expressions. These set of rules can be used to both reduce and simplify a complex boolean expression in an attempt to reduce the number of logic gates required. Some of the rules are:

Boolean law	Description
Complement Laws	$x \cdot \bar{x} = 0$ $x + \bar{x} = 1$
Law of the Double Complement	$\bar{\bar{x}} = x$
Idempotent Laws	$x + x = x$ $x \cdot x = x$
Identity Laws	$x + 0 = x$ $x \cdot 1 = x$
Dominance Laws	$x + 1 = 1$ $x \cdot 0 = 0$
Commutative Laws	$x + y = y + x$ $x \cdot y = y \cdot x$
Associative Laws	$x + (y + z) = (x + y) + z$ $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Distributive Laws	$x + (y \cdot z) = (x + y) \cdot (x + z)$ $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
DeMorgan's Laws	$\overline{x \cdot y} = \bar{x} + \bar{y}$ $\overline{x + y} = \bar{x} \cdot \bar{y}$
Absorption Law	$x \cdot (x + y) = x$ $x + (x \cdot y) = x$

Properties of Boolean algebra

Some Properties of Boolean Algebra are

- Boolean Algebra is defined in general by a set B that can have more than two values
- A two-valued Boolean algebra is also known as Switching Algebra. The Boolean set B is restricted to 0 and 1. Switching circuits can be represented by this algebra.
- The dual of an algebraic expression is obtained by interchanging + and · and interchanging 0's and 1's.

Boolean Algebra

- The identities appear in dual pairs. When there is only one identity on a line the identity is self-dual, i. e., the dual expression = the original expression.
- Sometimes, the dot symbol '·' (AND operator) is not written when the meaning is clear

Example 5.2.1: show that $A + A.B = A$

Solution

$A + A.B = A.1 + A.B$	identity
$= A.(1+B)$	distributive
$= A.1$	identity
$= A$	identity law

Example 5.2.2:

Simplify $(A+B).(A+C)$

$A.A + A.C + A.B + B.C$	– Distributive law
$A + A.C + A.B + B.C$	– Idempotent AND law (A.A = A)
$A(1 + C) + A.B + B.C$	– Distributive law
$A.1 + A.B + B.C$	– Identity OR law (1 + C = 1)
$A(1 + B) + B.C$	– Distributive law
$A.1 + B.C$	– Identity OR law (1 + B = 1)
$= A + (B.C)$	– Identity AND law (A.1 = A)

Thus $(A+B) \cdot (A+C) = A + (B.C)$.

Review Questions

1. If A and B are inputs to an Exclusive –NOR gate and C the output, show that the relation:
   $C = \bar{A}.\bar{B} + A.B$ exists.
2. Prove that De Morgan's laws are true with the use of truth table
3. Draw the logic circuit of the following expressions
   i. $X = \overline{A + B}$
   ii. $(A + B)(\overline{B} + C)$

❖ Differentiating between a compiler and interpreter

As stated earlier, both interpreter and complier translate a high level program to machine equivalent. However, the table below differentiates between compiler and interpreter.

Compiler	Interpreter
Compiler takes entire program as input	Interpreter takes single instruction as input.
Intermediate object code is generated	No intermediate object code is generated
Conditional control statements execute faster	Conditional control statements execute slower
Memory requirement is more (since object code is generated)	Memory requirement is less
Program needs not be compiled every time	Program has to be translated every time it is executed
Errors are displayed after entire program is checked	Errors are displayed for every instruction interpreted (if any)
Faster in execution	Slower in execution

6.2 Writing a computer program

Writing a computer program requires a careful planning otherwise the objective will be farfetched, the following steps are taken in writing a computer program:

- Define the problem
- Develop a solution
- Code the program
- Test the program
- Document the program

Step 1: Defining the problem

The task of defining the problem consists of identifying what it is to be done; the expected output, the required input to produce the expected output and process to

transform the input to the desired output. It should be able to analyse the problem in ind with:

- Thoroughly understanding the problem
- Understanding problem requirements
  - i. Does program require user interaction?
  - ii. Does program manipulate data?
  - iii. What is the output?
- If the problem is complex, dividing it into sub problems and analysing as above.

It is believed that with lucid description of the problem, half of the problem is solved.

Step 2: Plan the solution

Planning the solution is like the architectural design of a house. It depicts a finite sequence of precise steps to be taken in solving a given task. This finite sequence of steps is usually referred to as *Algorithm*. Algorithms describe actions to be executed and the order of execution of such actions. Two common forms of algorithm are *pseudocodes* and *flowchart*. However, in the modeling of object-oriented programs, UML (Unified Modeling language) diagrams are widely used.

Pseudocodes is an English-like nonstandard language that lets you state your solution with more precision than you can in plain English but with less precision than is required when using a formal programming language. Pseudo code permits you to focus on the program logic without having to be concerned just yet about the precise syntax of a particular programming language. Nevertheless, pseudo code is not executable on the computer. Consider these examples,

Example 1:

```
start
        Enter name
        IF name = "Oka" THEN
                OUTPUT "Why don't you invite Ube?"
        otherwise
                OUTPUT "Amala and ewedu can serve"
stop
```

Advantages	Disadvantages
translate the code as it is directly understood by the computer.	remembered.
	• It is hard to amend or find errors in a program written in the machine language.

6.1.2 Assembly Language

Assembly language was developed to overcome some of the many inconveniences of machine language and mostly used to help in compiler orientations. The instructions of the assembly language are converted to machine codes by a language translator and then executed by the computer. Some of the advantages and disadvantages are listed below:

Advantages	Disadvantages
• Assembly language is easier to understand and use as compared to machine language.	• Like machine language, it is also machine dependent / specific.
• It is easier to locate and correct errors than in machine language	• Since it is machine dependent, the programmer also needs to understand the hardware.
• It is easily modified compared to machine language	• Usually long and slow to write

6.1.3 High-Level Languages

High-level computer languages use formats that are similar to English. The purpose of developing high-level languages was to enable people to write programs easily, in their own native language environment. High-level languages are basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes. Each instruction in the high-level language is translated into many machine language instructions that the computer can understand.

Advantages	Disadvantages
High-level languages are user-friendly	A high-level language has to be translated into the machine language by a translator, which takes up time
They are similar to English and use English vocabulary and well-known symbols	The object code generated by a translator might be inefficient compared to an equivalent assembly language program
They are easier to learn	Slower to execute
They are easier to maintain	Difficult to manage libraries
They are problem-oriented rather than machine-based	Large amount of API's (Application Program Interface)
A program written in a high-level language can be translated into many machine languages and can run on any computer for which there exists an appropriate translator	
The language is independent of the machine on which it is used i.e. programs developed in a high-level language can be run on any computer	

❖ **Comparing high level and low level language:**

The following table compares high level language and low level language:

High-Level Languages	Low-Level Languages
Low development time	High development time
Easy to write and edit	Difficult to write and edit
Slower execution	Faster execution
Large file size	Small file size

Example 2: if a candidate wishes to be admitted into the university, the followi steps are required:

i. Register for UTME ;
ii. Write exam on the scheduled day
iii. IF score < 200 go to step 5
iv. Desperate to wear NYSC uniform? If yes, go to step 1 otherwise skip to step
v. Register for post UTME
vi. if post UTME score is greater than or equal to cut off
vii. smile home and come back for registration ; skip to step 9 otherwise
viii. check for other available options.
ix. end

Flowchart: a flowchart is a pictorial representation of step-by-step solution to problem. It is a map of what your program is going to do and how it is going to do it. It consists of arrows representing the direction the program takes and other symbol representing actions. The symbols used are standard set of symbols as developed b the American National Standards Institute (ANSI). There are two types of flowchart *systems flowchart* and *program flowchart.*

Systems flowcharts are basically used in systems analysis and design where they define major processing phases and data media used. Program flowchart on the other hand depicts the sequence of operations the computer will perform to solve a specific task. Some program flowchart symbols and what they are used for are shown in table 6.1.below

Oval	This is used to mark the terminal points in a Flowchart e.g. START and STOP
Rectangle	This is used for processing operations such as arithmetic calculations and assignment statements
Parallelogram	This is used for basic input and output operations
Decision	This box is used for decision making
Predefined Process	This is used to denote subprograms or Subroutines that can be called up within another program
Object Diagram / Onpage Connector	This denotes the point where one part of a flowchart connects to another on the same page
Offpage Connector	This denotes the point where one part of a flowchart connects to another on a differ
Behavior / Flowlines	These are used to designate the direction of flow of instruction

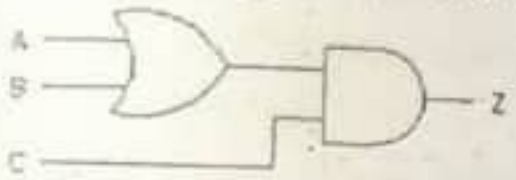
Table 6.1: Program Flowchart Symbols

Note:
Just as we have order of precedence in mathematics, boolean operators observe ord of precedence. The order of evaluation is:

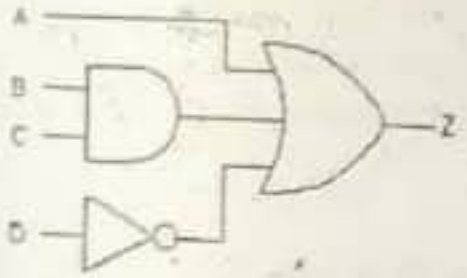
i. Parentheses
ii. NOT
iii. AND
iv. OR

Examples of logic gates diagram and truth table

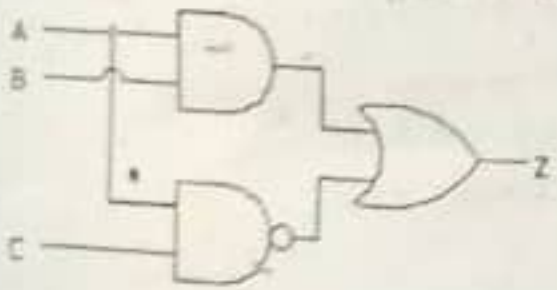
Example 5.1.1 Draw a logic circuit for Z= (A + B).C.



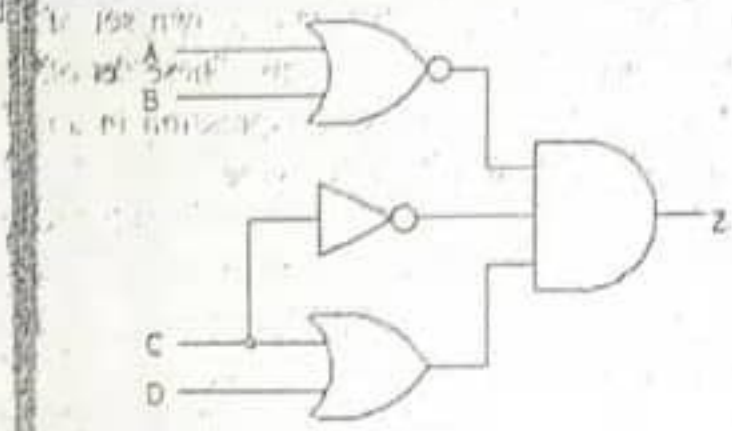
Example 5.1.2 Draw a logic circuit for $Z = A + B.C + \overline{D}$.



Example 5.1.3 Draw a logic circuit for Z= A.B + A.C.



Example 5.1.4 Draw a logic circuit for $Z = \overline{(A + B)}.(C + D).\overline{C}$.



Example 5.1.5 . If A = 1011 and B = 0100 are inputs to an exclusive-OR gate and C the output, show using the truth table that the relationship:
$C = A \oplus B = (\overline{A}.B) + (\overline{B}.A)$ exists

Solution:

A	B	$C = A \oplus B$	A	B	A.B	B.A	$C = (\overline{A}.B) + (\overline{B}.A)$
1	0	1	0	1	0	1	1
0	1	1	1	0	1	0	1
1	0	1	0	1	0	1	1
1	0	1	0	1	0	1	1

From the table, column 3 = column 8; thus the relationship $C = A \oplus B = (\overline{A}.B) + (\overline{B}.A)$ exists

Example 5.1.6: use the truth table to prove that $\overline{A + B} = \overline{A}.\overline{B}$

Solution

A	B	A+ B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

From the table, column 4 = column 7; this shows that $\overline{A + B} = \overline{A}.\overline{B}$ is true.

Rounded rectangle	This represents activities
Diamond	To represent decisions
(black circle)	This represents the start (initial state) of the workflow
(encircled black circle)	This represents the end (final state)
(horizontal bar) (vertical bar)	Horizontal or vertical bar indicate the start and end of concurrent threads of control.
(Arrows)	Arrows run from the start towards the end and represent the order in which activities happen.
(Constraints)	Action constraints are linked to an action in a node with text.

Activity diagram symbols

Properties of Algorithm

i. *Finiteness* – an algorithm must terminate after a finite number of steps have been taken
ii. *Unambiguous* - every instruction must be precisely described and clearly specified.
iii. *Sequence* of actions has unique initial action
iv. *Each action has a unique successor*
v. *Sequence of execution* - instructions are performed from top to bottom.

Program development

vi. Effectiveness- the solution prescribed is guaranteed to give a correct answer and that the specified process is faithfully carried out.
vii. Scope Definition - applies to a specific problem or class of problems.

Step 3: Coding the program

Coding is the translation of the architectural design of step 2 is translated into the syntax of a programming language such as visual C#.NET, JAVA, C++ and Visual Basic.NET; Programming statements are keyed in into the computer using the desired programming language.

Step 4: Testing and debugging

Testing entails running the executable file to see whether it produces the expected result. If not, debugging is carried out. Debugging means detecting, locating and correcting errors. There are three types of programming errors- syntax, run-time and logic errors. It can occur

- *Syntax error*: this type of error occurs when the rules of the language in use is violated such as incorrect punctuation, misspelt keywords and unbalanced parenthesis. Syntax errors are usually detected during compilation.
- *Runtime error*: this might result from a properly formed statement that pass the syntax checking, yet cannot execute properly. It can be as a result of trying to perform impossible arithmetic operations such as calculation with non numeric data, division by zero, computing the logarithm of a negative number or finding the square root of a negative number. This error is detected during execution of the program.
- *Logic error*: when a program is free from syntax and runtime errors but produced an erroneous result, such error is known as logic error. For example, if you are summing up ten positive integers and eventually the output yields a negative result, it shows that the program has logic errors.

Step 5: documenting the program:

Documentation is a detailed description of the program. It could be written within the program body as comments to improve readability and help others read and understand your program or as a separate document ("*manual*") specifying certain

facts about the program; such as logic tools, guide to the use of the program, ranges of the input data and minimum system requirements needed to run the program.

Documentation is needed to supplement human memory and to help organize program planning. It is also critical to communicate with others who have an interest in the program, especially other programmers who may be part of a programming team. Again, since turnover is high in the computer industry, written documentation is needed so that those who come after you can make any necessary modifications in the program or track down any errors that you missed.

Review Questions

1. What is a translator? Give the types of translators you know
2. What are the steps in writing a program?
3. Define an algorithm. What are the properties and what forms can it take.
4. What is debugging? State the types of error you can encounter during program development
5. What is (i) UML and what are the categories of UML diagram (ii) flowchart (iii) Pseudo code

b. Give the flowchart symbols and what they are used for

Chapter Seven

Introduction to Visual Basic.NET

0 Introduction

Visual Basic.Net originated from BASIC (Beginner's All Purpose Symbolic Instruction Code) which was designed as a language to teach people how to program by Professors John G. Kemeny and Thomas E. Kurtz at Dartmouth College New Hampshire in 1964. It was so successful that soon a lot of companies were using BASIC as their programming language of choice. Different versions of BASIC such as QBasic and GW basic evolved until in 1991 when Microsoft created Visual Basic 1.0. The main reason for the first version of Visual Basic was to make it a lot faster and easier to write programs for the new, graphical Windows operating system. This again was followed by enhanced versions every 1 or 2 years until the release of version 6.0 in 1998. Interestingly, these emerged versions were all *backward compatible*, meaning that later versions of VB could handle programs written with an earlier version.

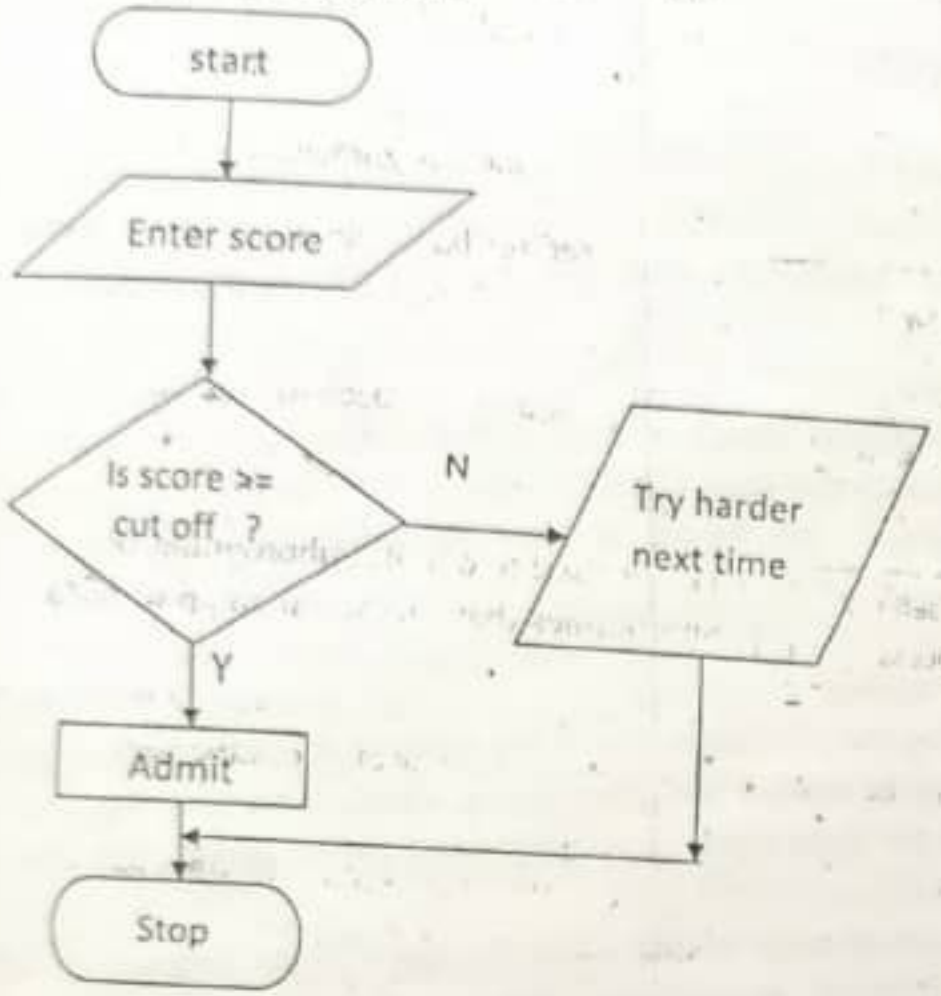
In 2002 however, Microsoft introduced Visual Basic .NET 1.0, a completely redesigned and rewritten version that was a key part to total computer software revolution at Microsoft. Because the .NET architecture was such a radical change, all programs written in Visual Basic 6.0 or earlier version had to be rewritten before they could be used with .NET. It was a controversial move at the time, but VB.NET has now proven to be a great programming language. One of the biggest changes in VB.NET was the use of object oriented software architecture; it should be noted that though VB6 was mostly *OOP*, VB.NET is totally OOP.

The Object concept

In VB, you will be working with ***objects***, which have ***properties***, ***methods*** and ***events***. Each object is based on a ***class***. These terms are explained briefly as follows:

- ***Objects*** : consider an object as a thing, or a noun. Examples include *forms* and *controls*. Forms are the windows and dialog boxes you place on the screen; controls are the components you place inside a form, such as text boxes, buttons and labels.

The figure is an example of a flowchart:



UML diagram

UML stands for Unified Modeling Language which is used in object on software engineering. Although typically used in software engineering, it is a language that can be used to model an application structures, behavior and business processes.

There are about 14 UML diagrams which can be divided into two main categor **structure diagrams** and **behavioral diagrams**. Figure 2 below gives the overvie the UML diagram categories.

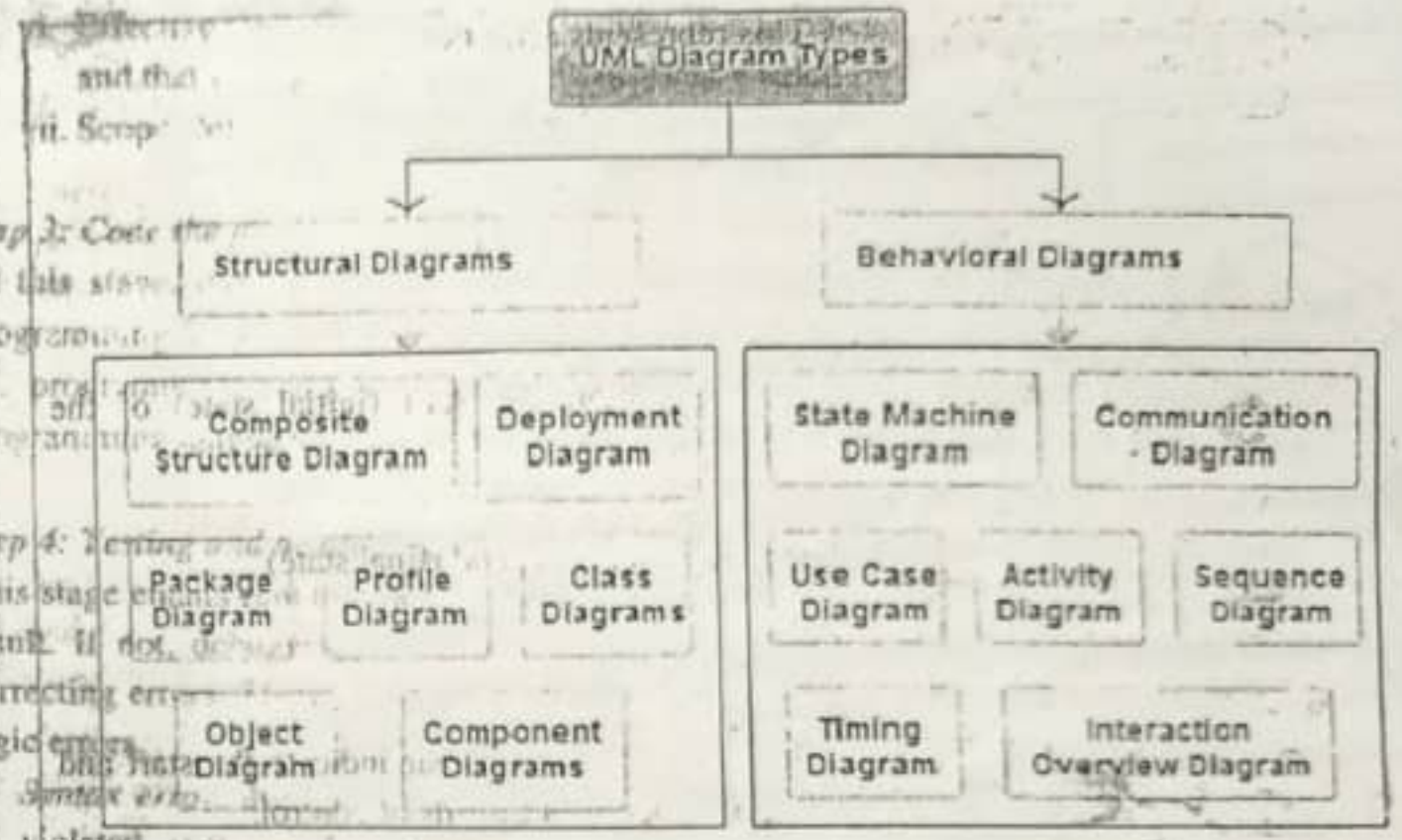


Figure.2 Types of UML diagram

Structural diagram: this shows the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavioural diagram: this shows the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

However, activity diagram which closely relates to the conventional flowchart will be utilized to illustrate some program structures in this text. Activity diagrams are typically used to model workflow or business processes and internal operations; and are constructed from a limited number of shapes, connected with arrows. Some UML activity diagram notations are depicted in table below. Other UML diagram notations are enclosed in the appendix B.

- *Properties*: they are attributes of an object. They give the description o
  object, such as colour, name, size and location. Thus they are like adject
  that qualify an object.
- *Methods*: these are actions associated with objects. They are the verb
  object-oriented programming. Examples include Close, Show, and Cl
  Each object has a set of methods that it can use. A method is like a proce
  or function written to implement operations.
- *Events*: an event occurs when the user takes an action, such as clickin
  button, pressing a key or closing a window. It can also be triggered by acti
  of other objects, such as a timer reaching a preset point. *Event handlers*
  procedures in your code that determine the action(s) to be performed when
  event occurs. When an event is *raised* (occurs), the event handler that recei
  the event is executed.
- *Classes*: a class is a template or blueprint used to create a new object. Class
  contain the definition of all available properties, methods and events. Ea
  time you create an object, it must be based on a class. For example, if y
  were to place three buttons on a form, each button is based on the Button cla
  and it is considered one object, called an *instance* of the class.

An analogy:

To drive home these concepts, let us consider an *Automobile class*. When we s
automobile, we are not referring to a particular auto, but we know that a
automobile has a *make* and *model*, *colour*, an *engine* and *number of doors*. Thes
elements are the **properties** of the Automobile class. Each individual *auto* is an
object, or an **instance** of the automobile class. Each automobile object has its
own settings for the available properties. For example, each object has a *Color*
property, such as *myAuto.Color = Blue* and *yourAuto.Color = Red*. The
methods, or actions of the automobile class could be *Start*, *SpeedUp*, *SlowDown*
and *Stop*. To refer to the methods of a specific object of the class, use
myAuto.Start and *yourAuto.Stop*. The **events** of an Automobile class could be
Arrive or *Crash*. **Procedures** are written to specify the actions you want to take
when a particular event occurs for an object. For example, a procedure for
yourAuto.Arrive event.

above terminologies, there are basic features of object oriented
include: *abstraction, encapsulation, in*

n

abstraction refers to the act of representing essential features without
nd details or explanations. Through the process of abstraction,
but the relevant data about an object in order to reduce
efficiency. In the same way that abstraction sometimes
that remains is a representation of the original, with unwanted
the concept of abstraction and are defined as a list of

Encapsulation: storing data and functions in a single unit is encapsulation. It is the
data and instructions inside an object. In general, encapsulation is
one thing within another thing so that the included thing is not
this means that it is the inclusion within a program object all the resources
function - basically, the methods and the data. Data cannot be
world and only those functions which are stored in the class

Inheritance: inheritance is the process by which objects can acquire the properties of
objects of other classes. In object-oriented programming, inheritance is the concept
objects is defined, any subclass that is defined can inherit the
of one or more of the general class. In other words, inheritance provides
additional features to an existing class without modifying it.
deriving a new class from the existing one. The new class will
combined features of both classes.

Polymorphism: polymorphism (overloading) means "one name, multiple forms";
that is the ability to take more than one form. An operation may exhibit different
behaviours at different instances. Specifically, polymorphism allows an entity such as
variable, a function, or an object to have more than one form. The behaviour of the
entity then depends on the data types used in the operation. For example, using
polymorphism we can create as many functions we want with one *function name* but
with different *argument list*. The function performs different operations based on the

- *Narrowing*: value of data is to be stored with lesser storage capacity. For example, converting double type to single.

They are also *implicit* or *explicit*, depending on the syntax in the source code.

- Implicit: an *implicit conversion* does not require any special syntax in the source code. VB automatically performs the conversion when the need arises. In the following example, Visual Basic implicitly converts the value of *num1* to a single-precision floating-point value before assigning it to num2.

```
Dim num1 As Integer
Dim num2 As Double
' Integer widens to Double.
        num1 = 24
        num2 = num1
```

- Explicit: an *explicit conversion* uses a type conversion keyword. Visual Basic provides several such keywords, which coerce an expression in parentheses to the desired data type. For example, in the following extension of the preceding example, the **CInt** keyword converts the value of num2 back to an integer before assigning it to num1.

```
ModuleModule1
SubMain()
Dim num1 AsInteger
Dim num2 AsDouble
' Integer widens to Double.
        num1 = 24    'assign 24 to num1
        num2 = num1  'assign num1 to num2
'num2 had been assigned the value 24 from num1.
        num2 = Math.Sqrt(num1)
        num1 = CInt(num2)
' num1 now has the value 5 (rounded square root of 24)
Console.WriteLine(num1)
Console.ReadKey()
EndSub

EndModule
```

Note: the program was written as ***Console Application***. Console applications, by default does not have graphical user interface (GUI) and are compiled into stand-

alone executable files thus keeping complexity at minimum. Hence forth, we shall use either Windows Application or Console Application for illustration.

Explanation of the program:

Line 1: `Module Module1` declares a module and the name of the module is *module1*.VB.Net is completely object oriented, so every program must contain a module of a class that contains the data and procedures that your program uses. Line 2 ***sub Main()*** defines the ***main procedure***, which is the entry point for all VB.Net programs. It states what the module or class will do when executed.

`num2 = Math.Sqrt(num1)` : finds the square root of *num1* and assigns it to *num2*.

`num1 = CInt(num2)` : Converts num2 to integer and assigns it to num1

`Console.WriteLine(num1)` : *WriteLine* is a method of the *Console* class defined in the *System* namespace. This statement causes the message (the value of *num1)* to be displayed on the screen.

The last line `Console.ReadKey()` : is to prevent the screen from running and closing quickly when the program is executed.

Some data type conversions functions are shown in the table below:

Functions	Description
CBool(expression)	Converts the expression to Boolean data type.
CByte(expression)	Converts the expression to Byte data type.
CChar(expression).	Converts the expression to Char data type
CDate(expression)	Converts the expression to Date data type
CDbl(expression)	Converts the expression to Double data type.
CDec(expression)	Converts the expression to Decimal data type.
CInt(expression)	Converts the expression to Integer data type.
CLng(expression).	Converts the expression to Long data type
CSByte(expression)	Converts the expression to SByte data type.
CShort(expression)	Converts the expression to Short data type.

floating point numbers with six digits of accuracy while Double stores double-precision floating point numbers with 14 digits of accuracy. Boolean can only store one of the two values- true or false, the Char data type can equally hold a single character such as k while String can hold alphanumeric data such as letters, digits and other characters.

User- Defined: as the name implies, it is defined by the programmer. Each member of the structure has a range determined by its data type and independent of the ranges of the other members.

Object: this can be used to store any type of data. If you fail to declare a variable, the variable will be seen as an object variable which can be used to store any value.

Data Type	Storage Allocation	Value Range
Object	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type Object

9.5: Declaring a variable

To declare a variable, the keyword *Dim* is used. It has the syntax:

Dim identifier As data type

Dim is indeed short for dimension (size). It tells the VB.NET to allocate memory to the new variable, whereas the As keyword is used to assign the type of data the variable will hold. For example,

```
Dim Regno As Integer              ' Regno is declared as an integer
Dim StudentName As String         ' StudentName is declared as string
Dim Salary As Double              ' Salary declared as double
Dim answer As Boolean             ' answer as boolean
Dim birthday As Date              ' birthday as date
```

Example 2: we can as well declare more than one variable using the same Dim statement. For example,

```
Dim count1, count2 As short       ' count1 and count2 are declared as short integer variables
Dim mtn, glo As Char              ' mtn and glo as char
```


Example 3: initial values can also be assigned to a variable during declaration as shown

```
Dim InitialValue As Single = 23.5
Dim Physical As String = " Computer"
```

Note: undeclared variables and variables declared without a data type are assigned the **Object** data type. This makes it easy to write programs quickly, but it can cause them to execute more slowly. However, specifying data types for all your variables is known as *strong typing*. Using strong typing has several advantages:

- It enables IntelliSense support for your variables. This allows you to see their properties and other members as you type in the code.
- It takes advantage of compiler type checking. This catches statements that can fail at run time due to errors such as overflow. It also catches calls to methods on objects that do not support them.
- It results in faster execution of your code.

9.6 Constant declaration

Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal. There are also enumeration constants.

To declare a constant, the keyword *Const* is used. It has the syntax:

Const identifier As datatype = value

For example,

Const pi *As Single* = 3.142 : where *pi* is the identifier and *3.142* is the value assigned to the identifier *pi*.

9.7 Type conversion

The act of converting a pre-defined data type to another is referred to as type conversion (*casting*). Conversions are either *widening* or *narrowing*, depending on the data capacities of the types involved.

- *Widening*: widening conversions preserve the source value but can change its representation. For example, converting *Single data type* to *double data type* (has enough room to accommodate the value)

For example, *GreAt* is the same as *great*, *GREAT*, *gReat*, and *Great*.
Some valid identifiers are:

Ahaneku, Unizik , csc101, csc102, ve_dvc, Igweeee, _total

Some invalid identifiers are:

5csc01	-	begins with a digit
Csc 101	-	contains special character (space)
Ada+obi	-	contains special character (+)
Friend	-	a reserved word
Stop	-	a reserved word

9.3 Constants

These are memory locations that hold data that cannot change during program execution. In other words constants refer to fixed values that the program may not alter during its execution. These fixed values are also called *literals*. Constants are treated just like regular variables except that their values cannot be modified after their definitions. There are numeric constants and string constants (string literals).

- Numeric constants may contain only digits (0…9), a decimal point, and a sign (+ or -) at the left side; special characters are not allowed. For example 786, 89.97, +23
- String constants may of course contain letters, digits and special characters. For example, "Unizik Post-UME exams: 12-09-15"

9.4: Data types

The data type of a constant or variable determines the type of information that will be stored in the allocated memory space. Data types can be grouped as primitive, user-defined and object data types.

Primitive data type: Visual Basic supplies a set of predefined data types, which you can use for many of your programming elements. This set of data types are referred to as *primitive* data types. These include:

Data Type	Storage Allocation	Value Range
Boolean	Depends on implementing platform	True or False
Byte	1 byte	0 through 255 (unsigned)

Char	2 bytes	0 through 65535 (unsigned)
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/- 79,228,162,514,264,337,593,543,950,335 (+/- 7.9...E+28) with no decimal point; 0 through +/- 7.9228162514264337593543950335 with 28 places to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)
SByte	1 byte	-128 through 127 (signed)
Short	2 bytes	-32,768 through 32,767 (signed)
Single	4 bytes	-3.4028235E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.4028235E+38 for positive values
String	Depends on implementing platform	0 to approximately 2 billion Unicode characters
UInteger	4 bytes.	0 through 4,294,967,295 (unsigned)
ULong	8 bytes	0 through 18,446,744,073,709,551,615 (unsigned)
UShort	2 bytes	0 through 65,535 (unsigned)

In summary, i. the memory in our computers is organized in bytes. A byte is the minimum amount of memory that we can manage in vb. A byte can store a relatively small amount of data: one single character. *Integer, long and short data types* can be used to store whole numbers such as *23456*. *Single* is used to store single-precision

Chapter Nine

Visual Basic Language Features

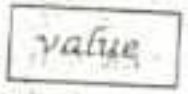
9.1 Character Set

We shall begin our discussion with characters recognized by Visual Basic. These include:

- Alphabetic characters: all the 26 letters of the English alphabet {A…Z, a…z}
- Numeric : the digitIs {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Special characters: [, & ( ) ^ % $ @ # " : _ ~ + / \ ! = * ' . <> …]

9.2 variables

The fundamental container in programming is a *variable*. Variables are memory locations that can hold data that do not change during program execution. Before a variable is used, it ought to be *declared*. When you declare a variable, Visual Basic reserves a memory location and assigns it a name called *identifier*.



a is the identifier and *value* is the content.

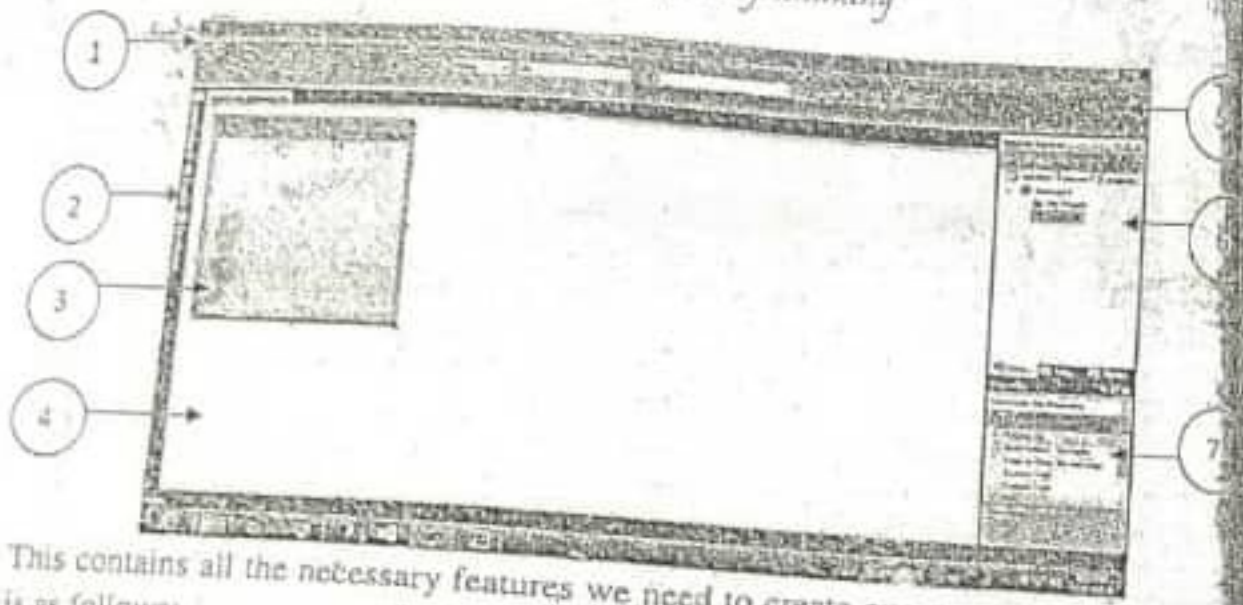
Identifier however, is a name used to identify a class, variable, function, or any other user-defined item. To form an identifier, the following rules are observed:

- An identifier must begin with a letter, followed by a sequence of letters, digits or underscore.
- The first character in an identifier cannot be a digit.
- It must not contain any special character such as) , #, : +.; however, an underscore (_) can be used.
- It should not be a reserved or keyword.

A *keyword* is any word that has special meaning to Basic. Some Visual Basic keywords are listed in the following table:

AddHandler	AddressOf	Alias	And	AndAlso	As	Boolean
ByRef	Byte	ByVal	Call	Case	Catch	CBool
CByte	CChar	CDate	CDec	CDbl	Char	CInt
Class	CLng	CObj	Const	Continue	CSByte	CShort
CSng	CStr	CType	CUInt	CULng	CUShort	Date
Decimal	Declare	Default	Delegate	Dim	DirectCast	Do
Double	Each	Else	ElseIf	End	End If	Enum
Erase	Error	Event	Exit	False	Finally	For
Friend	Function	Get	GetType	GetXML Namespace	Global	GoTo
Handles	If	Implements	Imports	In	Inherits	Integer
Interface	Is	IsNot	Let	Lib	Like	Long
Loop	Me	Mod	Module	MustInherit	MustOverride	MyBase
MyClass	Namespace	Narrowing	New	Next	Not	Nothing
Not Inheritable	Not Overridable	Object	Of	On	Operator	Option
Optional	Or	OrElse	Overloads	Overridable	Overrides	ParamArray
Partial	Private	Property	Protected	Public	RaiseEvent	ReadOnly
ReDim	REM	Remove Handler	Resume	Return	SByte	Select
Set	Shadows	Shared	Short	Single	Static	Step
Stop	String	Structure	Sub	SyncLock	Then	Throw
To	True	Try	TryCast	TypeOf	UInteger	While
Widening	With	WithEvents	WriteOnly	Xor		

It is worthy to note that visual Basic is not case sensitive. This means that upper case and lower letters are considered to be identical unlike in some programming languages such as Java and C++.



This contains all the necessary features we need to create our programs. The labeli
is as follows:

1. Menu bar
2. Toolbox
3. Form designer
4. Document window
5. Toolbar
6. Solution explorer
7. Properties window

Description of the labeled items:

- *Menu bar and toolbar*: these bars are similar to those that most window applications use. The *menu bar* contains command for managing the IDE an for developing, maintaining and executing programs. The set of command when selected cause the IDE to perform specific action. For example, save file, open a window, print a file, and so on. The *toolbar* on its part contain graphics (*icons*) that graphically represent command. Rather than navigating the menus from the menu bar, you can access many of the more commo commands from the toolbar. They serve as shortcut to frequently used commands.
- *Form Designer*: this is where the form that makes up your user interface is designed. By default, when a new project is created, a form is added to the project with the name Form1. You can then change the name of the form as desired.

- *Document window:* this is the largest window in the center of the screen. Most of the work on your application is done here; we can place or rearrange controls on a form, edit the source code, and so on.
- *Toolbox*: this contains the icon representing controls you can place on the form. We can either drag and drop these controls on the form or double click a control in other to place it on the form.
- *Solution Explorer*: this shows a collapsible view of all the files that are part of your application including a list of the classes it references.
- *Properties window*: this displays the properties of the currently selected object in the design view. Properties specify information about the object such as size, colour or position.

Note: If you desire to use a window such as solution explorer, toolbox or properties window and it is not visible, simply click the icon on the tool bar or on the *view* menu and select them.

Having examined these we are ready to write our first project. Just relax ...and you will be happy oh.

Writing your first program:

We can continue with the form above or we can start a new project by following the procedure below:

- click on File menu
- select New
- click on Project

A window will be launched, select Visual Basic, Windows Form Application and enter a name in the name field. For the name, enter *MyFirstProject* and click OK. We are back to the IDE again. *You see...what goes around comes around.*

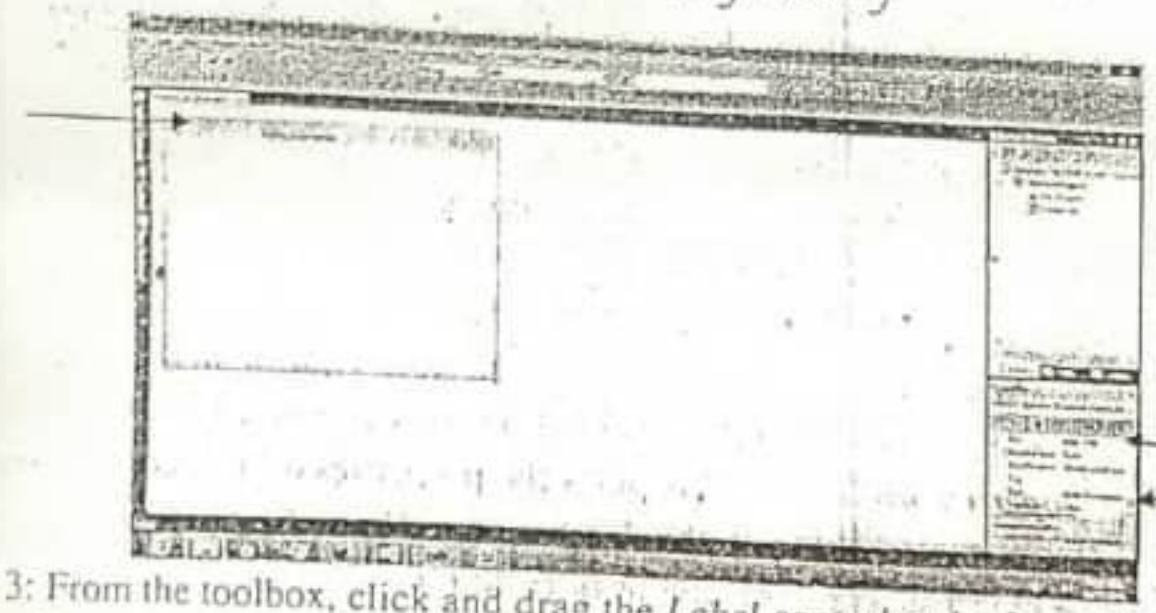
Step2: click on the form to highlight it. Set the following properties at the properties window:

Text : Hello Everyone

Size : 450, 350

Backcolor: Info

and press the enter key. This will cause form's title bar, the size and background colour to be updated immediately as shown below.



Step 3: From the toolbox, click and drag the *Label control* to the form. The label control is generally used to display some informative text on the GUI which is not changed during runtime.

Step 4: set the following properties for the label control in the property window:

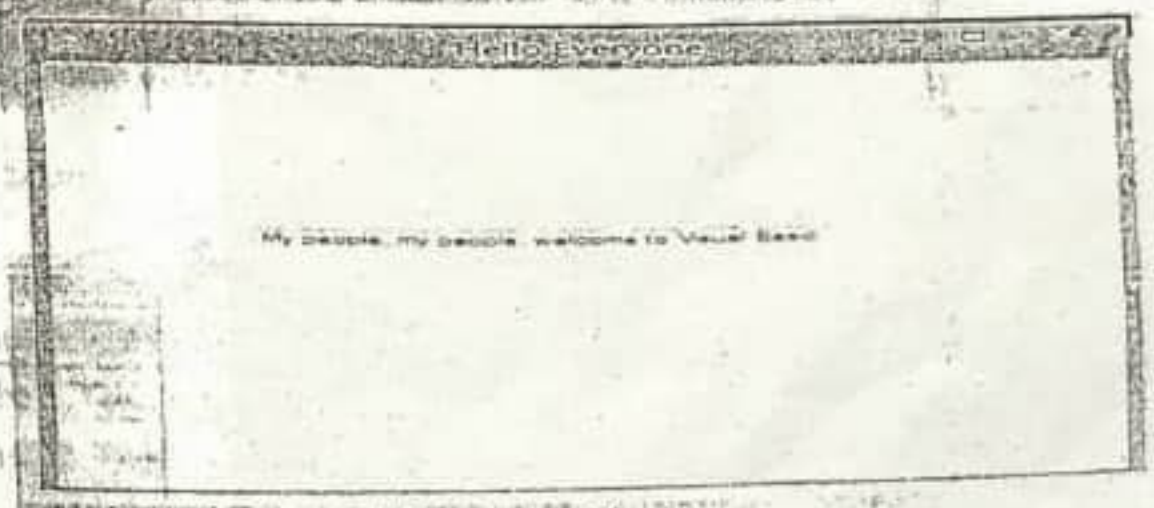
i. *Autosize property: True* (this allows the label to resize itself and accommodate the text)
ii. *BorderStyle: Fixed3D*
iii. *Text: My people, my people, welcome to Visual Basic*

Some Label properties and description are as follows:

S/N	Property	Description
i.	**Autosize**	Gets or sets a value specifying if the control should be automatically resized to display all its contents.
ii.	**BorderStyle**	Gets or sets the border style for the control.
iii.	**Font**	Gets or sets the font of the text displayed by the control.
iv.	**FontHeight**	Gets or sets the height of the font of the control.
v.	**ForeColor**	Gets or sets the foreground color of the control.
vi.	**PreferredHeight**	Gets the preferred height of the control.
vii.	**PreferredWidth**	Gets the preferred width of the control.
viii.	**TabStop**	Gets or sets a value indicating whether the user can tab to the Label. This property is not used by this class.
ix.	**Text**	Gets or sets the text associated with this control.
x.	**TextAlign**	Gets or sets the alignment of text in the label.

Step 5: save the project by selecting **File menu>Save All**

Step 6: Run the project by pressing F5 or select **Debug** from the menu bar and choose **Start Debugging**. This will cause the project to execute and output as follows



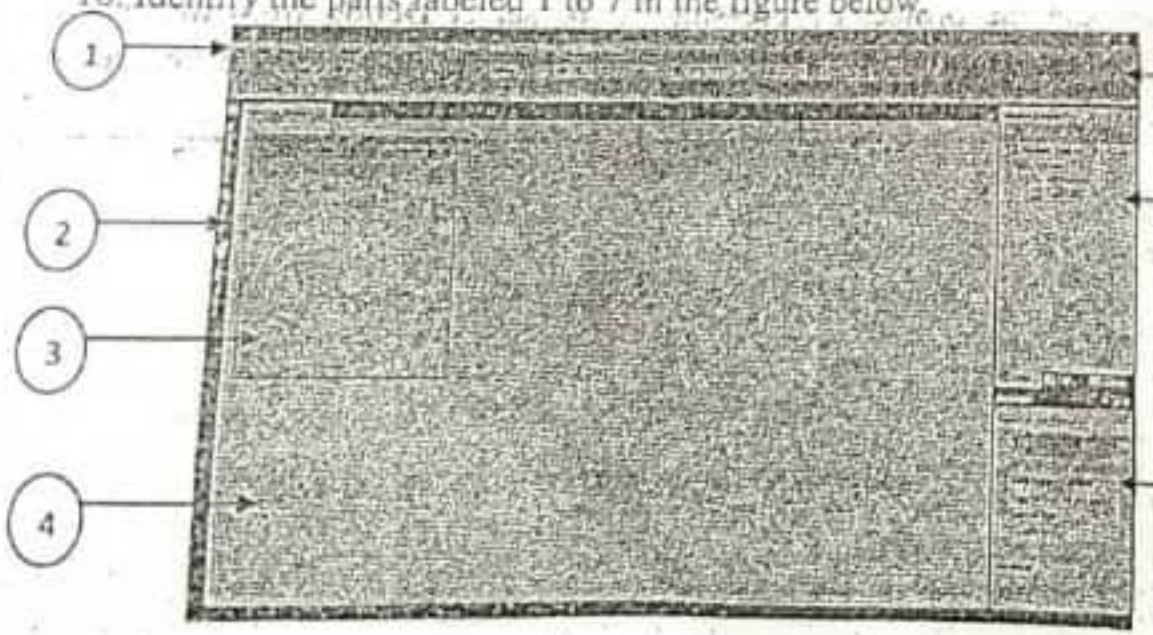
And we are done. It was not difficult, was it?

7.4 Design time, Run time and Debug time

Visual Basic has three distinct modes. While you are designing the user interf
writing codes, you are in ***design time***. When you are testing and runninn
project, you are in *run time*. However, if you get a run-time error or pause
execution, you are in ***debug time***. The IDE title bar indicates *Running* or *Def*
to show that a project is no longer in design mode.

Review Questions

1. Who invented BASIC and in what year?
2. BASIC stands for ----------
3. What is an object, class, properties, methods and events.
4. Mention any 5 programming languages supported by the .NET framewo
5. List any six types of application you can develop with Visual Basic.NET
6. What are the functions of the following: document window, toolbox, s
   explorer, properties window and form designer
7. In what 2 ways can you place a control on a form
8. Give any 5 attributes of a form
9. What are the three modes of Visual Basic
10. Identify the parts labeled 1 to 7 in the figure below.



Chapter Eight

Adding More Controls to the Form

Introduction

as many as desired controls on a form such as more forms, labels,
xes. Let us now illustrate by adding more controls to our existing

2.Text

Example 8.1: 2 button controls to the form (a button control is generally used to
event by providing a handler for the Click event) and set the text

Button1 = Action1
Button2 = Action 2

Label1 = "" (empty)

the following properties for *Label1*

Autosize property = False. (This will enable us to resize the label to our desired size)

Backcolor = Custom> choose green from the palette

TextAlign = MiddleCenter (this will cause the text to be positioned in the middle of the label control)

Double click on Button1, (this will take you to the ***code window*** where you can place the code that will execute when this button is clicked) and enter the following code:

Label1.Text = "Heaven is REAL"

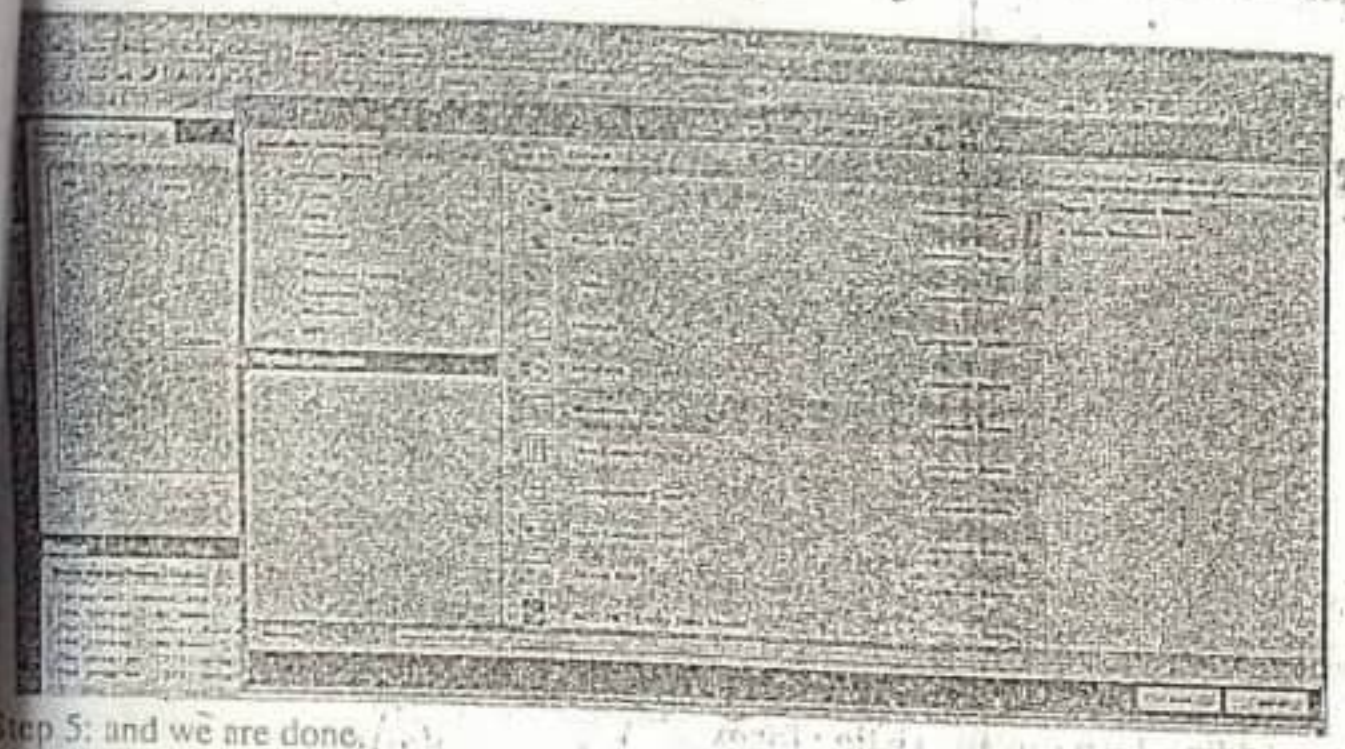
Explanation of above

Label1 is the name of the label control. You can change the name in the Name field of the properties window when you highlight it. The dot (.) that follows is a convention to adding a property to an object (***name the object, add a period and then name the property***).*Heaven is Real* is the text that will be displayed without the double-quotes.

Again, double-click the Button2 control and enter the following code

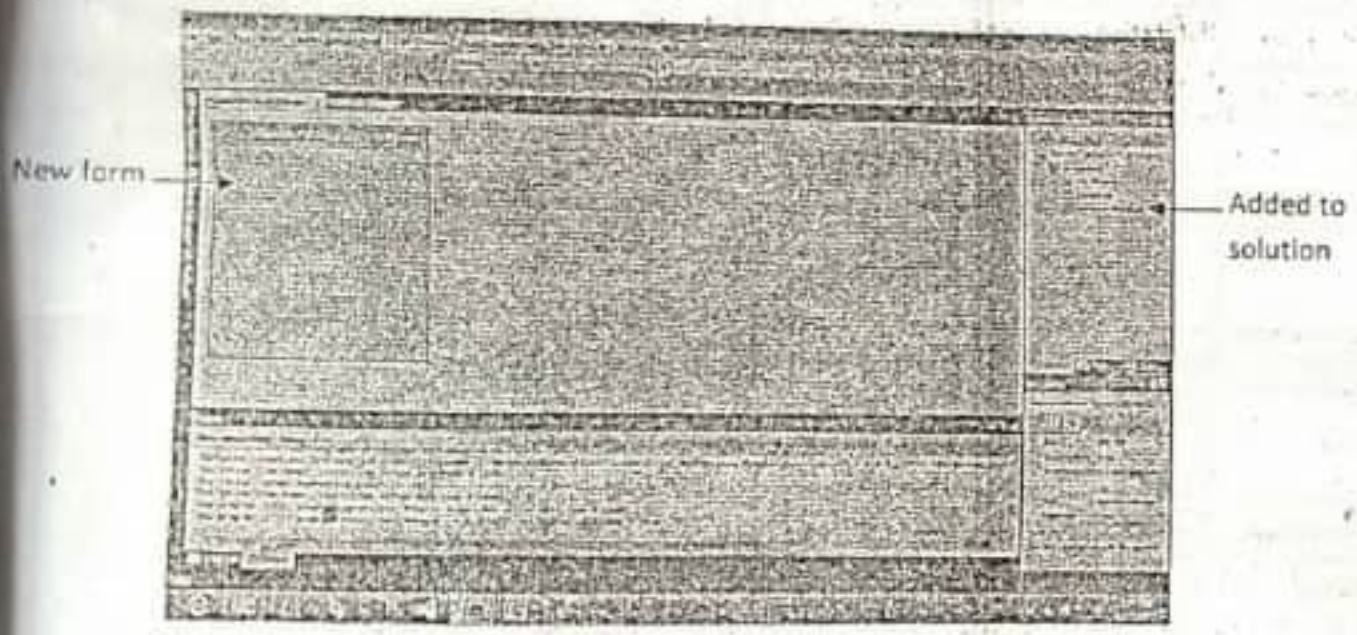
Label1.Text = "Hell is REAL"

The explanation is as in Button1 above.



Step 5: and we are done.

The new form will now be displayed on the screen and be added to the Solution Explorer.



Review Questions

What are dialog boxes used for? Give the DialogResult enumerations you know Write a program that will display (i) Ube na Oka: Egwu Eji (ii) Heaven helps those that help themselves. Using Message boxes.

Create a project and name it Exercise 1and set the forms properties as follows:

Text = Just a simple one. Size = 435, 327. BackColor = light blue.

ii. Add a *Label* control and set it as follows:

AutoSize = False. Location = 58, 132. Size = 307, 30. Font = Lucidn handwriting. backColor = Silver. fontStyle = ItalicBold. FontSize = 10

Save and run the project

4. Create the calculator GUI shown below as follows:

i. Create a new project named *Exercise Two.*

ii. Set the form's properties as follows:

Size = 272, 192. Text = Calculator. Font = Tahoma

iii. Add a *Text box* and set the properties as follows:

Text = 0. Size = 240, 21. TextAlign = Right . Location = 8, 16

iv. Add *panel* to the form to contain the calculator's numeric keys. Panel controls are used to group other controls. Set the properties as follows:

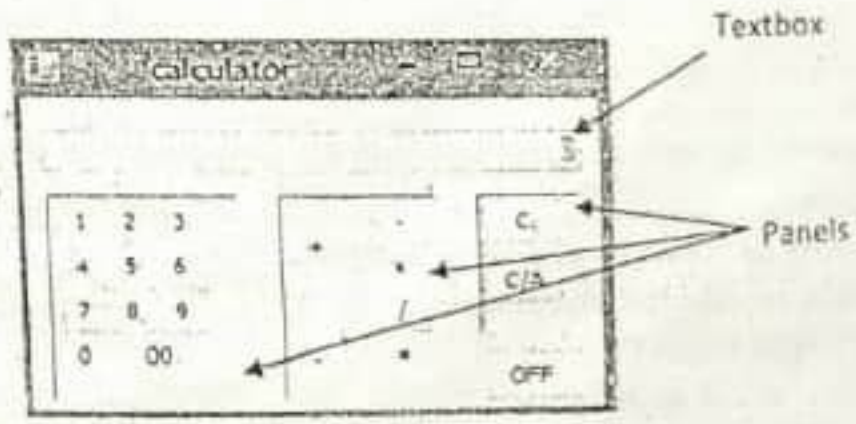
BorderStyle = Fixed3D. size = 88, 102. Location = 8, 48.

Add a second panel to contain the calculator's operator keys with the following properties: *BorderStyle = Fixed3D. size = 72, 102. Location = 112, 48.*

v. Add the 3rd panel to contain the calculator's C (clear) and C/A (clear all) keys. Set it as follows: *BorderStyle = Fixed3D. size 48,72. Location = 200, 48*

vi. Add Buttons (20) to the form. Change the text properties as shown in the figure (1, 2,…., C/A, …., +, OFF). Each of the buttons should be set with size: 24,24. 00 and OFF with size 46, 24. The + buuton 24, 48 and C and C/A with size 34, 24.

vii. Save and Run the project.



All of these dialog box control classes inherit from the **CommonDialog** class and override the *RunDialog()* function of the base class to create the specific dialog box. The RunDialog() function is automatically invoked when a user of a dialog box calls its *ShowDialog()* function.

The **ShowDialog** method is used to display all the dialog box controls at run-time. It returns a value of the type of **DialogResult** enumeration. The values of DialogResult enumeration are:

- **Abort** - returns DialogResult.Abort value, when user clicks an Abort button.
- **Cancel**- returns DialogResult.Cancel, when user clicks a Cancel button.
- **Ignore** - returns DialogResult.Ignore, when user clicks an Ignore button.
- **No** - returns DialogResult.No, when user clicks a No button.
- **None** - returns nothing and the dialog box continues running.
- **OK** - returns DialogResult.OK, when user clicks an OK button
- **Retry** - returns DialogResult.Retry , when user clicks an Retry button
- **Yes** - returns DialogResult.Yes, when user clicks an Yes button

8.2 Displaying output with the MessageBox

Let us illustrate the use of dialog box with the message box. *MessageBox* is one of these **built-in dialog boxes** that help you to provide a rich user interface in your front-end applications. This dialog box lets you to display **custom messages** to your users and accept their **input** regarding the **choice** that they have made.

Example 8.3.1: write a program that will display ***Programming in VB is fun, what do you think?*** using a MessageBox.

Solution

Again, we start by designing our GUI. Let us simply use a button control for this purpose; the message ***Programming in VB is fun, what do you think?*** will be displayed if we click on the button control at run time.

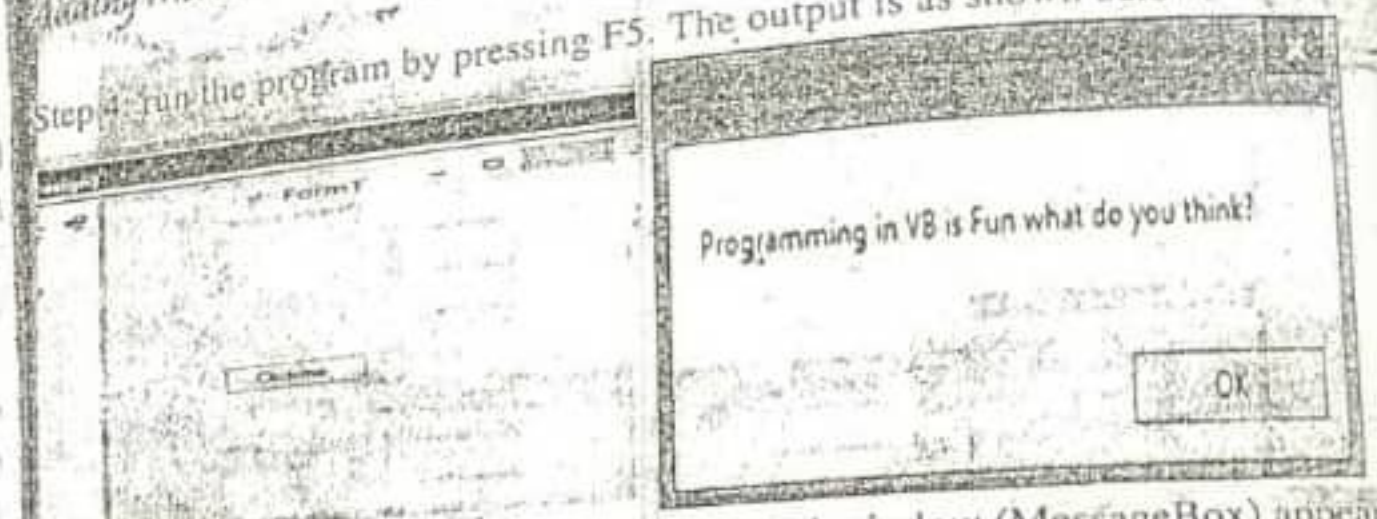
Step 1: place a button control on a form and set the Text property to ClickMe.

Step 2: double click on the button to open the code window

Step 3: enter the following code:

```
MessageBox.Show("Programming in VB is Fun what do you think?")
```


Step 4: run the program by pressing F5. The output is as shown below.



When the ClickMe button is clicked, the second window (MessageBox) appears. Click OK to return to the design mode:

8.3 Adding forms to an existing form.

In reality, a project can contain more than one form. To add form to an existing for simply follow the following steps:

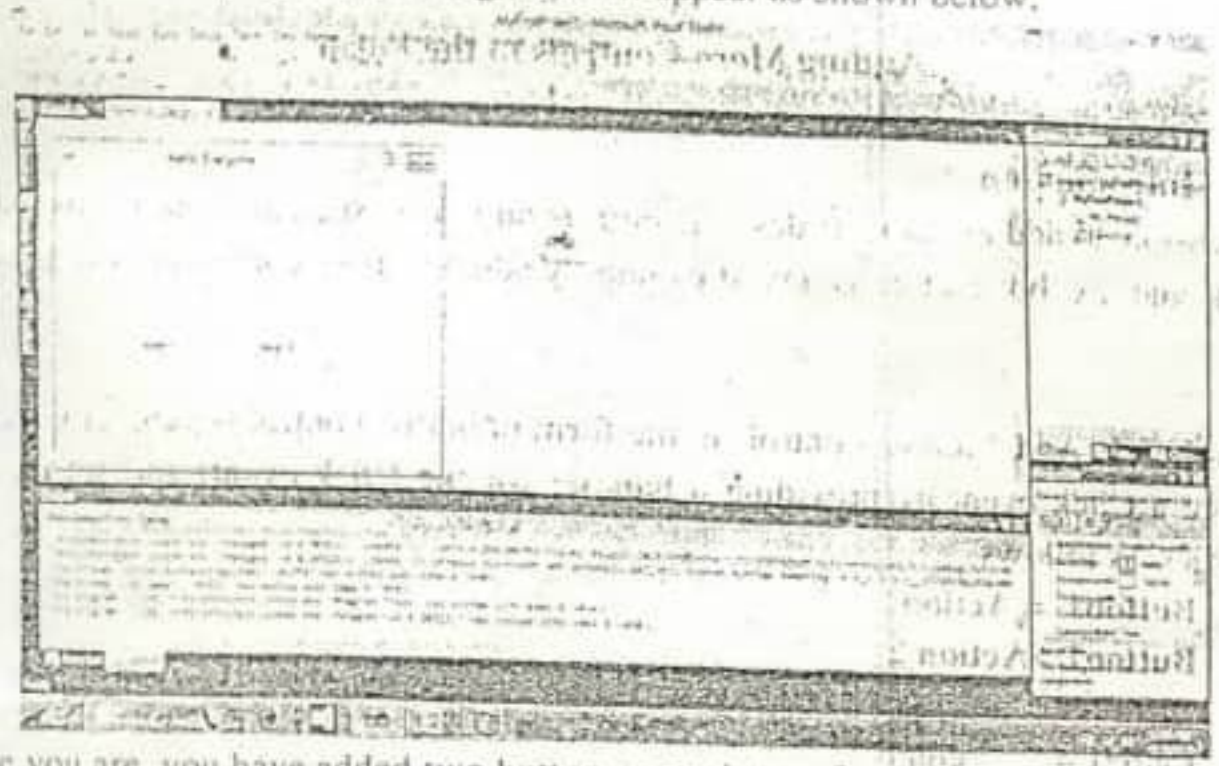
Step 1: select *Project*

Step 2: select *Add Windows Form*

Step 3: in the *Add New Item* dialog box select *Windows Form* from the template l

Step 4: Enter a name for the new form say ***MySecondForm*** and click *Add*

When you run the program, a dialog box will appear as shown below:



There you are, you have added two button controls to a form. If you click on Action 1 Heaven is Real is displayed in the label control and when you click on Action 2, Hell is Real will be displayed.

Example 8.2: write a program that adds two numbers using the TextBox control.

Solution:

Step1: we need a new form, so we start by creating a new project. We then insert two textboxes, three labels and one button. The two text boxes are for the user to enter two numbers, one label is to display the addition operator, another to display the equal sign and the last label is to display the result.

Step 2: Set the properties as follows:

Label1 Text = + ; Label2 Text = '=' ; Label3 Text = " " (empty) and

Button1 Text = ADD

Double click on Button1 control to open the code window and enter the following codes:

```
Dim num1, num2, result As Single
num1 = TextBox1.Text
num2 = TextBox2.Text
result = num1 + num2
Label3.Text = result
```


Explanation:

Dim num1, num2, result As Single :

The above statement declares 3 variables *num1* , *num2* and *results* as floating-point variables of single data type.

num1 = TextBox1.Text : this assigns the value the user enters in TextBox1 to num1

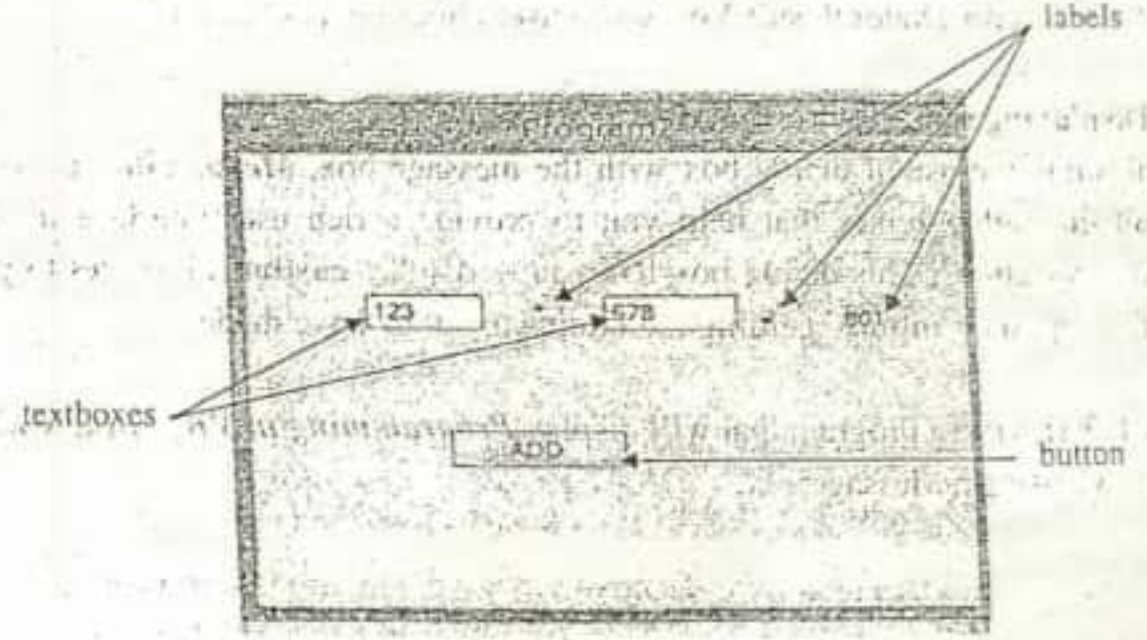
num2 = TextBox2.Text : this assigns the value the user enters in TextBox2 to num

result = num1 + num2 : add values in num1 and num2 and store this in result

Label3.Text = result : assigns the result to Label3 Text property.

A bit confusing..., don't worry we shall explain all these later and the picture will become clearer oh.

Now, **run the program.** The output of the program is as shown below:



8.1: Dialog boxes

There are many built-in dialog boxes to be used in Windows forms for various tasks to the user of an application such as opening and saving files, displaying messages, printing a page, providing choices for colours, fonts, page setup and so on. These built-in dialog boxes reduce the developer's time and workload.

argument list in the function call. The exact function to be invoked will be determin
by checking the *type* and *number of arguments* in the function. Polymorphism
extensively used in implementing Inheritance.

In summary, VB.NET is a simple, modern, object-oriented computer programmin
language that combines the power of .NET Framework and the common langua
runtime with the productivity benefits that are the hallmark of Visual Basic. The .N
framework applications are multi-platform applications. It has been designed in su
a way that it can be used from any of the following languages: Visual Basic, C#, C+
Jscript, and COBOL, etc. to develop variety of applications such as Windo
applications, Web applications and Web services. More so, all these languages c
access the framework as well as communicate with each other.

7.1 Programming in VB.NET

To program in VB.NET, you need to first install the software. Download and insta
VB.NET Express Edition from Microsoft from https://www.visualstudio.com/e
US/products/visual-studio-express-vs. Click on the download button, which will no
deposit the installation file on your system. The latest version is 2015 edition but f
the illustrations in this text, 2010 edition will be used.

7.2 The visual studio.NET environment.

After the installation, it is time to run it. By clicking on the icon, for example 2010
noted with: the program is launched and the start page will appear thus:

Step 2: Choose *New project*. This will launch the window that displays list of available programming languages supported by .NET framework such as visual c#, visual F# and of course Visual Basic alongside the type of applications that can be developed with them as shown below:

Step 3: click on Visual Basic and the list of available applications that can be developed with it ranging from *windows forms applications*, *ASP.NET web forms application* to *console application* will be displayed.

Step 4: choose Windows Forms Application and enter the name you wish for your project in the name field. Click OK to launch the VB IDE in the *design view* as shown below:

CSng(expression).	Converts the expression to Single data type
CStr(expression)	Converts the expression to String data type.
CULng(expression)	Converts the expression to ULng data type.
CUShort(expression)	Converts the expression to UShort data type.

Review Questions

1. Define (a) variable (b) constant (c) identifier (d) keyword
2. What are the rules of forming a valid identifier
3. Identify invalid identifiers in the following and state why they are considered invalid:
   (i) Friend (ii) Praise God (iii) 2_number (iv) _MgtSc (v) Object (vi) LawFaculty (vii) one+one (viii) Bus Stop (ix) UNIziK (x) APCPdP
4. Mention the data types that can store whole numbers. (b) what data type can be used to store a single character
5. Declare a variable called *AdmissionStatus* and assign to it *Admitted, Congratulations!!!*
6. What do you understand by implicit and explicit type conversion
7. The statement *f=CInt(b)* will do what?

Chapter Ten

- Visual Basic Operators

10.0 Introduction

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. Visual Basic.Net supports the following operators:

- Arithmetic Operators
- Unary Operators
- Relational Operators
- Logical/Bitwise Operators
- Bit Shift Operators
- Assignment Operators
- Concatenation operators

10.1 Arithmetic Operators:

These are operators for performing arithmetic operations. The table below shows the arithmetic operators supported by VB. In the example, assume X = 25 and Y = 3

Operator	Name	Description	Example
^	Exponentiation	Raises one operand to the power of another	X^Y = 15625
+	Addition	Adds two operands	X + Y = 28
-	Subtraction	Subtracts second operand from the first	X - Y = 22
*	Multiplication	Multiplies both operands	X * Y = 75
/	Division	Divides one operand by another and returns a floating point result	X/ Y = 8.33333
\	Integer division	Divides one operand by another and returns an integer result. It truncates the fractional part of the division	X / Y = 8
MOD	Modulus	Returns the remainder after an integer division	X MOD Y = 1

Note:

1. When floating point numbers are used with integer division operator, the number is first rounded to the next whole number and then divided. For example, if *a,b,c,d* were declared as *double*, then
   - i. a = 24.4 \ 5 will produce 4 : rounded to 24
   - ii. b =24.6\5 will produce 5 : rounded to 25
   - iii. c = 24.4 / 5 will produce 4.88
   - iv. d =24.6 / 5 will produce 4.92
2. When there are integer and floating numbers in an expression involving division, the result is rounded up if it is to be assigned to an integer. For example, if *r* and *t* were declared as *integer* and *x, y* as *double* then
   - i. r = 24.4 / 5 will produce 5
   - ii. t = 24.6 / 5 will produce 5
   - iii. x = 24.4 /5 will produce 4.88
   - iv. y = 24.6 / 5 will produce 4.92

10.2 Unary operators:

Unary operator takes only one operand. VB provides the unary operators (+) and (-) so that programmers can write expressions such as +89 and -756.

10.3 Relational Operators

This compares two expressions and returns a Boolean value that represents the result of the comparison. The relational (comparison) operators are shown in the table below: assume A is 56 and B is 34

Operator	Description	Example
=	Equality	(A = B) will return False
<>	Inequality (not equal to)	(A <> B) will return True
>	Greater than	(A > B) will return True
<	Less than	(A < B) will return False
>=	Greater than or equal to	(A >= B). will return True
<=	Less than or equal to	(A <= B) will return False

10.4 Logical / Bitwise Operators

The table below shows the logical and bitwise operators. Assume variable A is True and variable B False, then:

Operator	Description	Type of operator		Example
And	If both operands are true, then condition becomes true otherwise it is false	logical	Bitwise	(A And B) is False.
Or	Returns True If either or both of the operands is True and returns false only when all the operands are false	logical	Bitwise	(A Or B) is True.
Not	Gives the inverse of the logical state of its operand. If a condition is true, then Logical NOT operator will make it false and if false will make it true	logical	Bitwise	Not A is False.
Xor	Logical Exclusive OR operator. It returns True if both expressions are True or both expressions are False; otherwise it returns False.	logical		A Xor B is True.
AndAlso	It is logical AND operator. It works only on Boolean data. It performs short-circuiting.	logical		(A AndAlso B) is False.
OrElse	It is the logical OR operator. It works only on Boolean data. It performs short-circuiting.	logical		(A OrElse B) is True.
IsFalse	It determines whether an expression is False.	logical		
IsTrue	It determines whether an expression is True.	logical		

Note: i. *short-circuiting* means that if *expression1* is True, then *expression2* is not evaluated.

i. Bitwise operators supported by VB.Net are *And, Or, Xor and Not.*

Operator	Example: assume A = 12 and B= 30	Equivalent
=	C = A	C = 12
+=	A += 5	A = A+ 5 : 17
-=	A -= 5	A = A – 5 : 7
*=	A *= 5	A = A*5 : 60
/=	B /= A	B = B / A : 2.5
\=	B\= A	B = B\A : 2
^=	A ^=2	A= A ^ 2 : 144
<<=	B <<= 2	B = B << 2 : 120
>>=	B >>= 2	B = B >> 2 : 7
&=		Str1 &= Str2 is same as Str1 = Str1 & Str2

10.8 Operators Precedence

Operator precedence determines the order in which terms in an expression is evaluated just the same way we apply BODMAS in mathematics. Operators with higher precedence are executed before others in an expression. For example, the multiplication operator has higher precedence than the addition operator. In the table below, operators with the highest precedence appear at the top of the table and those with the lowest appear at the bottom:

Operator	Precedence
Exponentiation (^)	Highest
Unary identity and negation (+, -)	
Multiplication and floating-point division (*, /)	
Integer division (\)	
Modulus arithmetic (Mod)	
Addition and subtraction (+, -)	
Arithmetic bit shift (<<, >>)	
All comparison operators (=, <>, <, <=, >, >=)	
Negation (Not)	
Conjunction (And, AndAlso)	
Inclusive disjunction (Or, OrElse)	
Exclusive disjunction (Xor)	Lowest

Review Questions

1. Mention the operators supported by VB.NET
2. Identify invalid identifiers from these statements:
   (i) P ^= K + 2 (ii) 2k = m mod n (iii) write = judge + tribunal
   (iv) stress = pressure + fatigue (v) GoTv = DSTV – supersports
3. If a = " Bukola " , b= " Saraki" and c = " senate president" then what is :
   (i) Z = a & b & c (ii) k = c + " for the 8th " + b (iii) W = b + a
4. Perform the following bit shift operations:
   (i) 22 >> 2 (ii) 120 << 3 (iii) 5 >> 2 (iv) 89 >> 2
5. Show the order of execution of these expressions: (i) $y = ax^2 + bx + c$ (ii)
6. If p = 4 , r = 35 , q = 5 , w = 12 , x = 2 and y = 3; evaluate: (i) y ^ x (ii) q \ x (iii) p Mod w (iv) w /q (v) z = p *r Mod q + w / x – y (vii) x <> y (vii) r >= w/x (viii) p^ x*2 = x Mod y * w

Chapter Eleven

Basic Statements

11.0 Introduction

A statement is a complete instruction in Visual Basic programs. It may contain keywords, operators, variables, literal values, constants and expressions. Statements could be grouped as declaration and executable statements:

Declaration statements - these are the statements used in naming or defining a variable, array, constant, or procedure, and as well specify a data type.

Executable statements - these are the statements that initiate actions. These statements can call a method or function, loop or branch through blocks of code or assign values or expression to a variable or constant. Example is an assignment statement.

11.1 Remark statement (comment):

Comments can be used to document what the program does and what specific blocks or lines of code do. Visual Basic compiler ignores comments; hence you can include them anywhere in a program. To insert a comment, you can use any of the following methods:

i. Type an apostrophe followed by the comment. For example,

```
'This is a comment on a separate code line.
sum = a + i          'assign a+b to sum
```

ii. Precede the comment with REM. For example,

```
REM This is another comment on a separate code line.
MessageBox.Show("smile, Jesus loves u") REM displays
enclosed message
```

iii. Select the lines and click on the *Comment* or *UnComment* button on the IDE toolbar.

iv. Shortcut: you can use [*ctrl +k, ctrl +c*] to comment and [*ctrl +k, ctrl +u)* to uncomment

11.2 Displaying multiple statements on a line.

Multiple statements can be written on a line separated by a *colon* (:) character. For example,

```
Dim sampleString As String = "Hello World":
MessageBox.show(sampleString)
```

11.3 continuing a statement over multiple lines

A statement usually fits on one line, but when it is too long, you can continue it onto the next line using a line-continuation sequence, which consists of a *space* followed by an *underscore* character (_), followed by *enter* key. In the following example, the MessageBox executable statement is continued over two lines.

```
Public Sub demoBox()
    Dim nameVar As String
    nameVar = "John"
    MessageBox.show("Hello " & nameVar _          'line 1
                    & ". How are you?")           'line 2
End Sub
```

11.4 Input/ output statements

To enter data into the system, *input statements* are used. Some input commands used in VB console applications are: *Console.Readline* , *Console.Read* and *Console.ReadKey*. To display information to the user output statements are used. Some output commands in VB console applications are: *Console.Write* and *Console.Writeline*

The following program illustrates these:

```
Module Module1

    Sub Main()
        Dim Name As String
        Console.Write("Please type your name: ")
        Name = Console.ReadLine()
        Console.WriteLine("Hello " + Name)
        Console.ReadLine()
    End Sub

End Module
```

Sample output of the above program is

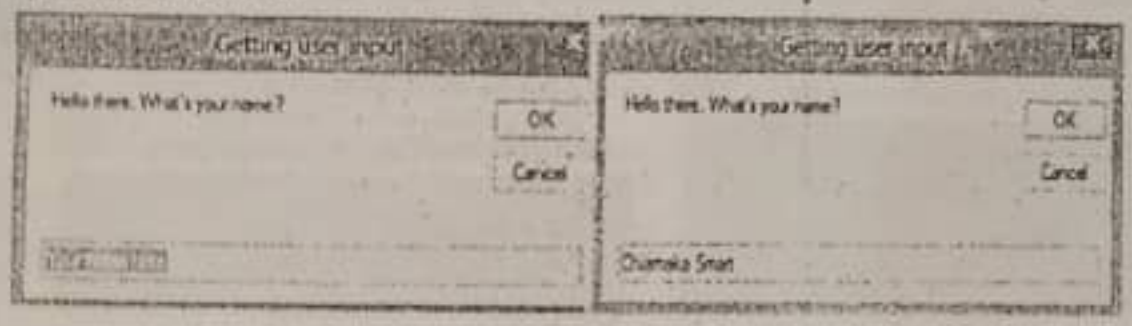
file:///c/users/e/documents/visual studio 2010/Projects/ConsoleApplication4/

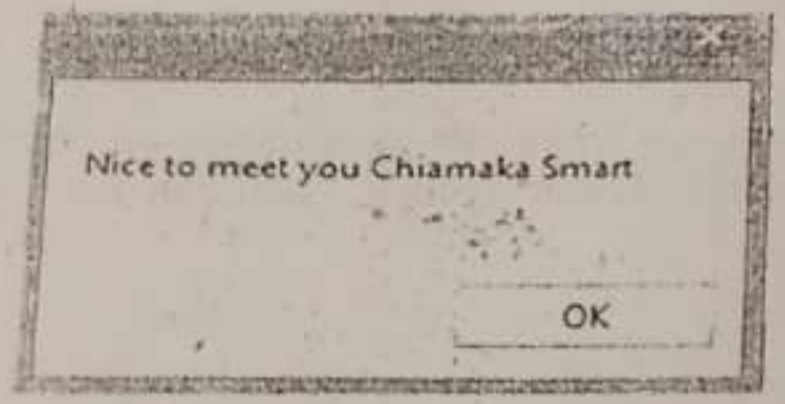
```
please enter your name Asogwa Doris
Hello dear Asogwa Doris
```

Example 2: the following program uses InputBox() and MessageBox() to accept and display information to the user on windows application

```
    Dim prompt As String = String.Empty
    Dim title As String = String.Empty
    Dim defaultResponse As String = String.Empty
    Dim answer As String
prompt = "Hello there. What's your name?" ' .
title = "Getting user input"
defaultResponse = "Your name here"
    ' Display prompt, title, and default value.
answer = InputBox(prompt, title, defaultResponse)
    ' Say something to the user
    MessageBox.Show("Nice to meet you " & answer)
```

The outputs are as shown:





Explanation:

The first 4 lines were used to declare the variables: prompt, title, defaultresponse and answer. The first three set their initial values to null (empty string). Lines 5,6 and 7 assign values to *promt*, *title* and *defaultResponse* respectively. Line 8 is a comment and ignored by the compiler. Line 9 is an input statement with the keyword *InputBox*. This consists of 3 parameters: the first *prompt* will contain the message to be displayed in the dialog box, the second *title* is the string expression that will be displayed on the title bar of the dialog box and the third *defaultResponse*is the string expression displayed on the text box as the default value if no other input is provided. Line 11 is a *MesageBox* that will display the string expression on quote with the input entered by the user.

Review Questions

1(a) What is a comment statement? (b) How do you comment a block of statements. (c) What commands can you use to comment a line of statement? (d) Give the short cut keys to comment and uncomment a block of statements

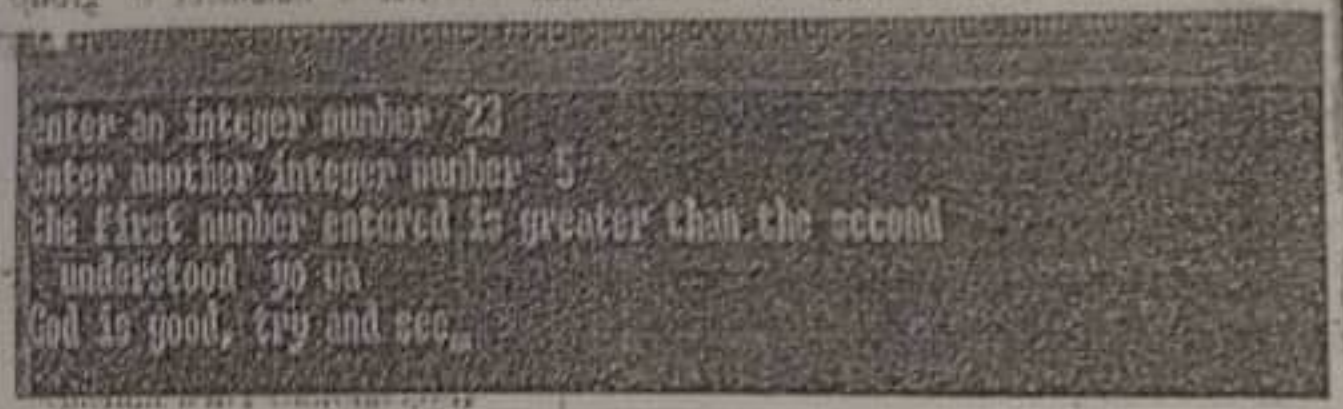
2 What operator can you use to (a) display multiple statements on a line (b) continue a statement over multiple lines

3 what is the output of this statement:
Console.Writeline("Smile Jesus") : console.Write (" loves you")

4 Write a statement to display the following: ***Minister without portfolio*** on the screen

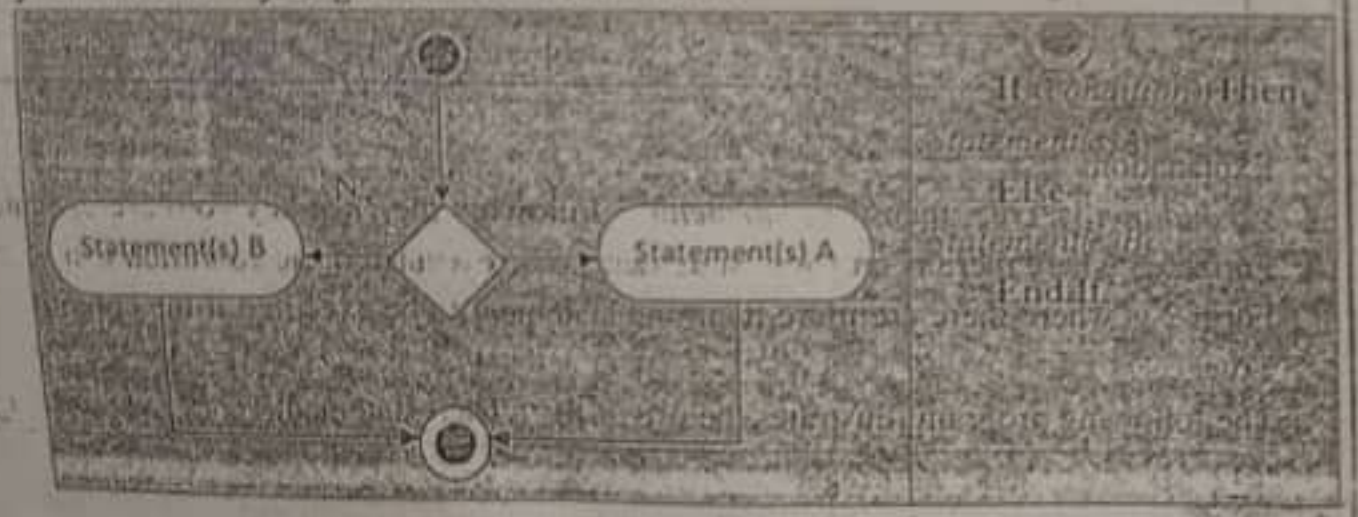
5. Use a MessageBox to display the following statement: " *No shortcut to Heaven*"

Sample output of the program is as shown:



12.2.2 The If ...Then...Else statement

This allows alternative action to be taken when the condition is false. That is to say that when the condition is true, a statement or sequence of statements are executed and when it is false, another statement or group of statements are executed. The syntax and activity diagram are as shown below:



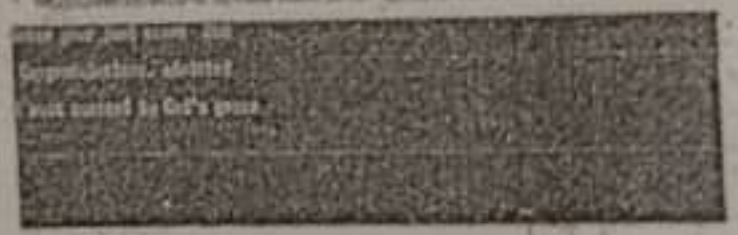
Explanation:

If the condition is true then *statement(s) A* will be executed otherwise *statement(s) B* will be executed. In either case, only *statement(s) A* or *statement(s) B* will be executed and not both. The program below illustrates this:


```
Dim JambScore As Integer
Console.Write("Enter your Jamb Score: ")
  JambScore = Console.ReadLine()
  If JambScore >= 200 Then
Console.WriteLine()
Console.WriteLine(" Congratulations, admitted")
Else
Console.WriteLine(" sorry, not admitted try harder next time")
EndIf
Console.WriteLine()
Console.Write("I must succeed by God's grace")
Console:ReadKey()
```

Explanation;

The second line displays *Enter your Jamb Score*. Line 3 accepts the score entered by the user. Line 4 tests the score against 200 and if *true*, displays *Congratulations, admitted* afterwhich control is taken to the statement following *End if*. If condition is *false*, *sorry, not admitted try harder next time* is displayed. Again control is transferred to the statement following *End if*. Sample outputs are follows:



case 1: True

case 2: false

12.2.3 Nested If....Then.....Else

If... Then...Else can also be used for multiple test conditions and actions. In this case, If...Then... Else is embedded inside another if ...then...else. This is referred to as nesting. It has the syntax:

```
If (condition1) Then
Statement(s)A
 Else If (condition2) Then
Statement(s)B
Else If (condition 3)Then
        Statement(s) C
....
Else    Statement(s) N
    End If
```

Explanation:

f condition1 is true *statement A* is executed and control is transferred to the tatement following *End if* otherwise the second condition is tested and if true *tatement B* is executed and control transferred to the statement following *End If* otherwise condition 3 is tested and so on. If none of the conditions is true then the *lse* part is executed. Consider the following program that assigns grade to score.

```
Dim Score AsInteger
Console.Write("Enter your score ")
      Console.WriteLine()
Score = Console.ReadLine()
      Console.WriteLine()
If Score >= 70 Then
Console.WriteLine(" Grade is  "&"A")
ElseIf Score >= 60 Then
Console.WriteLine(" Grade is  "&"B")
ElseIf Score >= 50 Then
Console.WriteLine(" Grade is  "&"C")
ElseIf Score >= 45 Then
Console.WriteLine(" Grade is  "&"D")
ElseIf Score >= 40 Then
Console.WriteLine(" Grade is  "&"E")
Else
Console.WriteLine("Grade is   "&"F")
EndIf
Console.WriteLine()
Console.Write("programming is easy...")
Console.ReadKey()
```


When the above program executes, the system will request for a score. The score is then tested against the multiple options starting with the first until a condition is satisfied and necessary action taken. If none of the conditions is met, the last else part is executed.

Sample outputs of the above program are as follows:

12.2.4 Select...Casse statement

A **Select Case** statement allows a variable to be tested for equality against a list of values. It is an alternative to If...Then...Else for selectively executing one block of statements from among multiple blocks of statements. A Select Case statement provides capability similar to the If...Then...Else statement, but it makes code more readable when there are several choices.

A Select Case structure works with a single test expression that is evaluated once at the top of the structure. Visual Basic then compares the result of this expression with the values for each Case in the structure. If there is a match, it executes the block of statements associated with that Case. It has the following syntax and activity diagram shown in *figure xx*.

The *expression* of the *select case* is evaluated and the value is tested against *expression a*, if there is a match, *statement(s) a* will be executed. If there is no match, the value is tested against *expression b* if there is a match, *statement(s) b* will be executed and if not testing continues with other case expressions. However, if none of the case expressions is true, the case else *statement(s)* will be executed. Note that the *expression of the selectcase* must evaluate to any of the elementary (primitive) data types such as Boolean, Integer, Char, Double, Date and Byte.

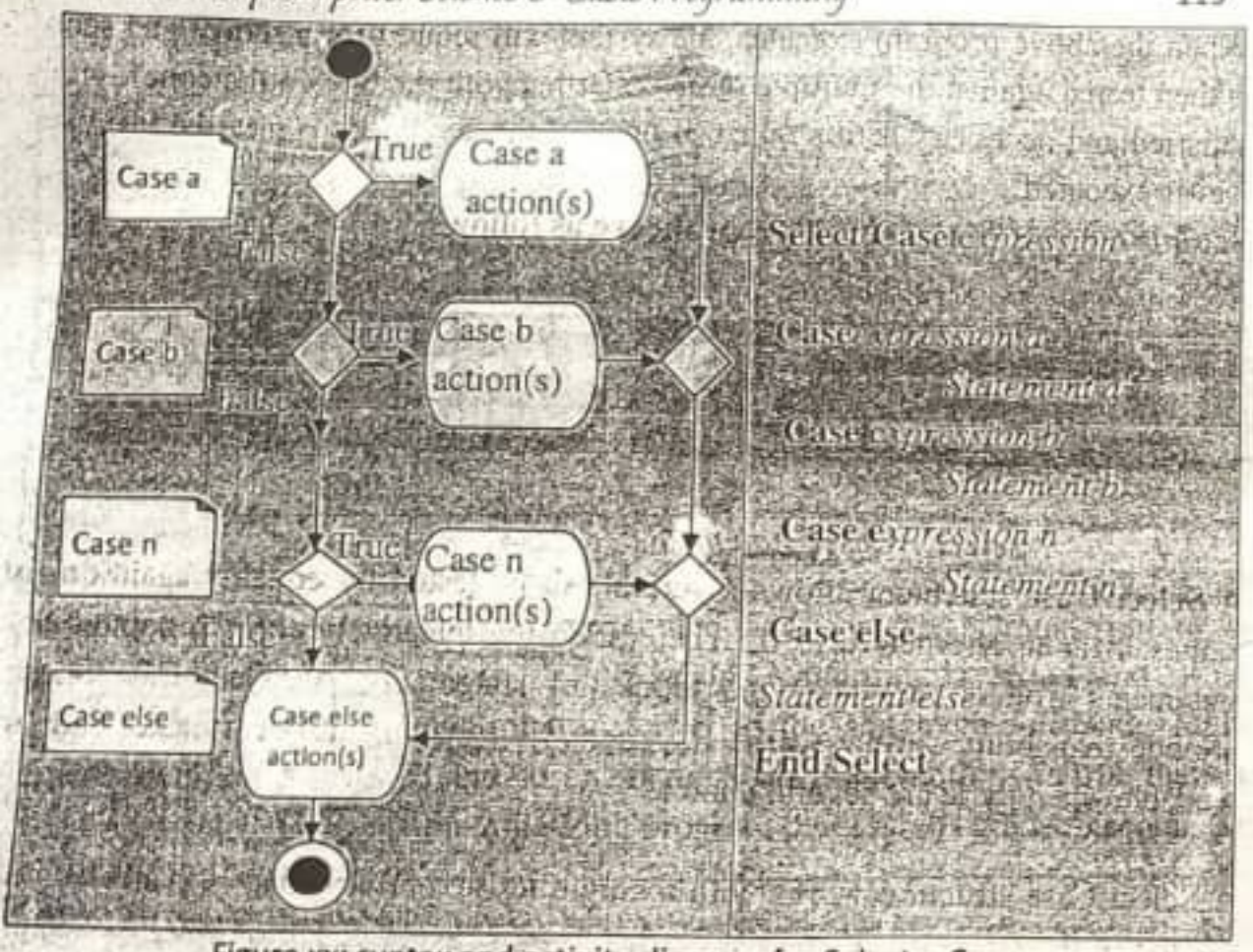


Figure xx: syntax and activity diagram for Select ...Case

e following program illustrates the select case statement:

```
Dim score AsInteger
Dim grade AsChar
Console.WriteLine("Enter  your score")
score = Console.ReadLine()
SelectCase score
Case 70 To100 : grade = "A"
Case 60 To69 : grade = "B"
Case 50 To59 : grade = "C"
Case 45 To 49
                grade = "D"
Case 40 To44 : grade = "E"
Case 0 To39 : grade = "F"
Case Else :Console.WriteLine("invalid score")
EndSelect
Console.WriteLine(grade)
Console.ReadLine()
```


Sample outputs of the program are as shown:

case1: invalid score

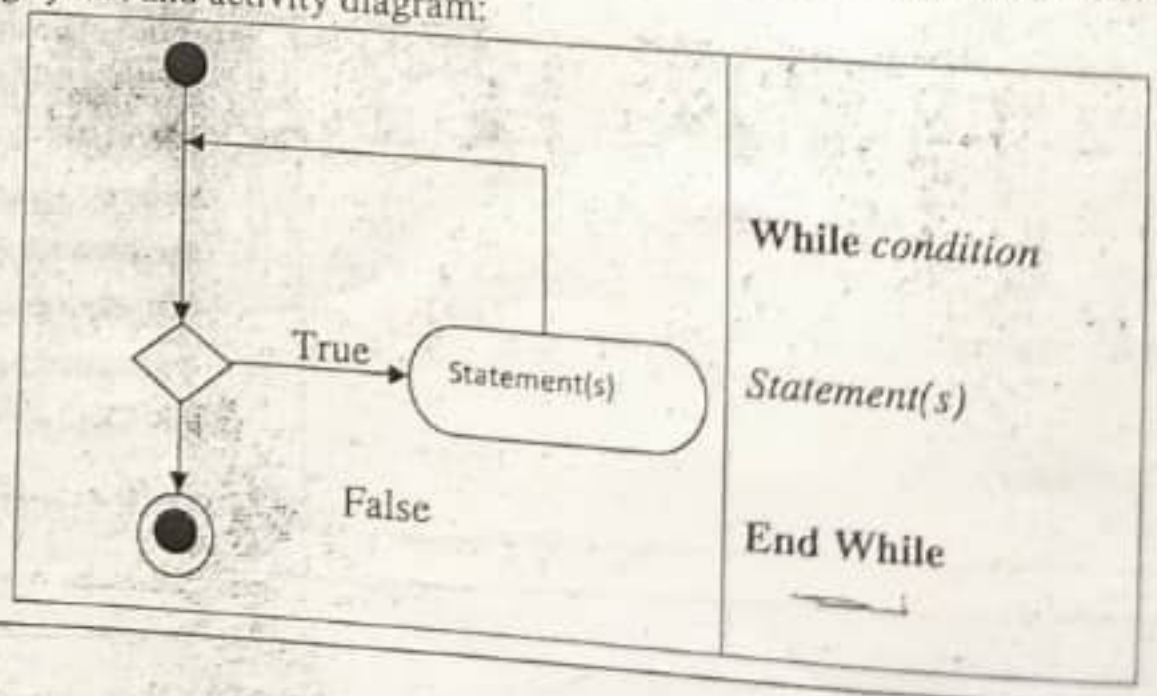
Case 2: score of 78

12.3 Repetition (iteration)

Whereas selection statements let you execute a section of a code based on a certain conditon, repetition statement (also called a looping statement) allows you to specify an action that should be *repeated* depending on the value of a condition. The act of executing the same section of code multiple times is referred to as *looping*. Visual Basic provides the following repetition statements: the While, Do while...Loop, Do ...Loop While, Do Until ...Loop, Do ... Loop Until and For ...Next.

12.3.1 The While statement

The While statement executes a section of code while a condition is true. It has the following syntax and activity diagram:

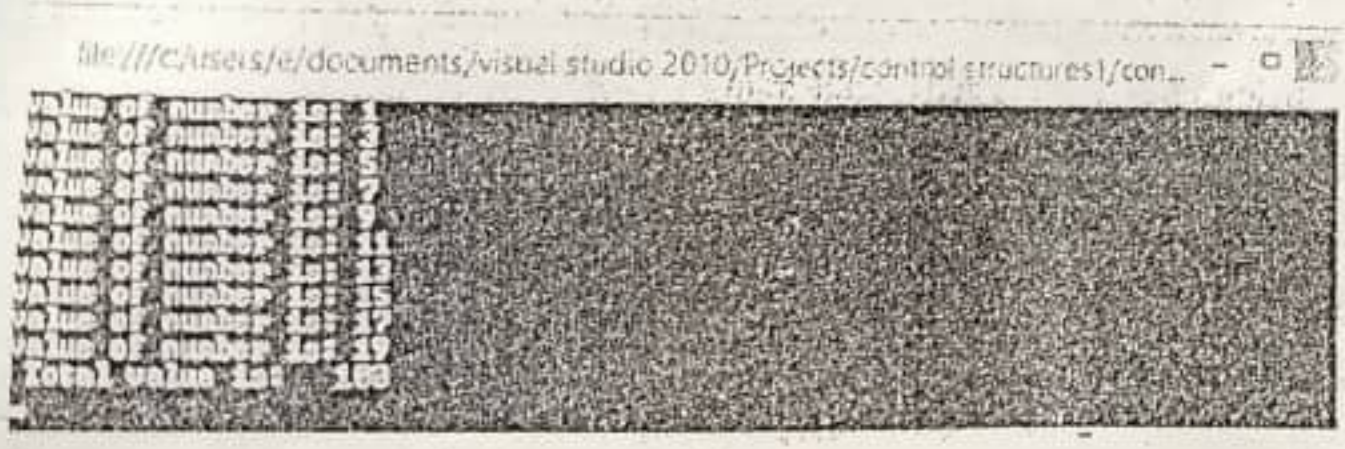


Explanation:

While the *condition* remains *true*, *statement(s)* will be executed. When the condition is *false*, the statement following *End While* will be executed. The statement(s) contained in the *While* statement constitute the *body* of the while. The following program illustrates the While structure:

```
SubMain()
Dim sum AsInteger = 0
DimnumAsInteger = 1
While (num) <= 20
sum = sum + num 'add num to d current value of sum
Console.WriteLine("value of number is: "&num)
num = num + 2
EndWhile
Console.WriteLine(" Total value is:    "& sum)
Console.ReadLine()

EndSub
```

The above program uses While statement to display and accumulate the odd integers between 0 and 20. Line 4 tests the value of *num* and as long as it is less than or equal to 20; lines 5,6 and 7 will be executed. The moment line 4 becomes false, line 9 will be executed which displays the final (accumulated) sum. Below is the output of the program:

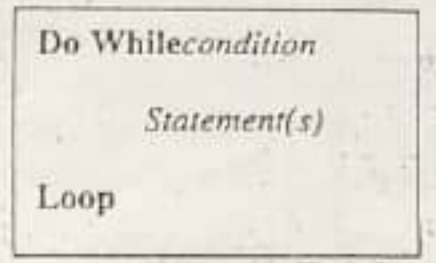


odd integers b/w (0 and 20)

12.3.2 Do While...Loop

The *Do While ...Loop* statement behaves like the *While* statement discussed earlier. The only variation is the inclusion of *Do* in the *While* and use of *Loop* in place of *End While*.

The activity diagram is same as the *While* and the syntax is as follows:



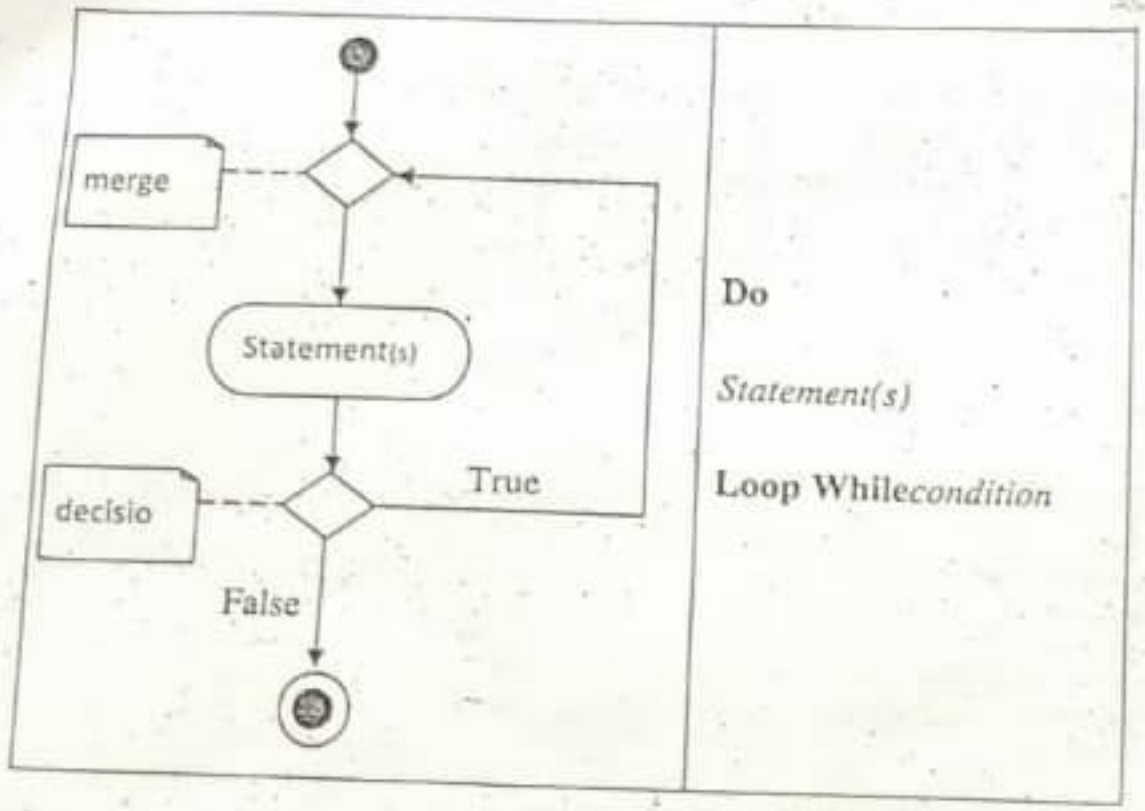
As an example, let us rewrite the above program with Do While...Loop.

```
SubMain()
Dim sum AsInteger = 0
DimnumAsInteger = 1
Do While (num) <= 20
sum = sum + num 'add num to d current value of sum
Console.WriteLine("value of number is: "&num)
num = num + 2
Loop
Console.WriteLine(" Total value is:    "& sum)
Console.ReadLine()

EndSub
```

The output is same as that obtained with the *While*.

12.3.3 Do ...Loop While

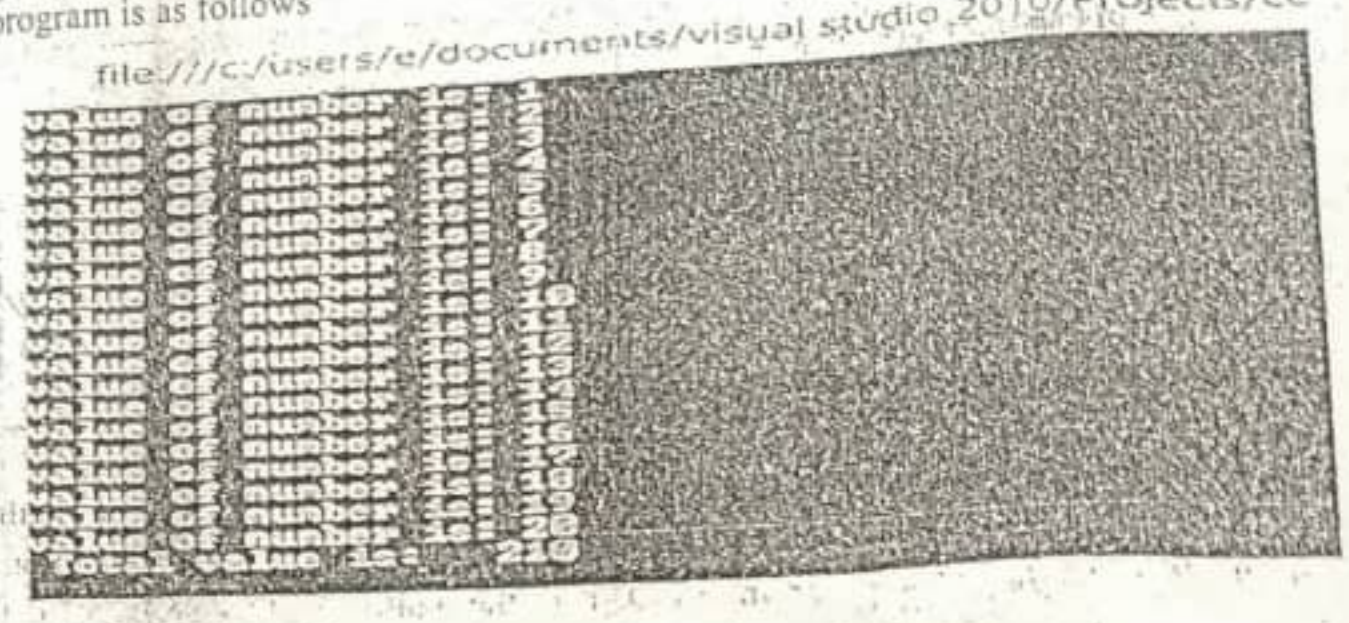
The Do Loop ...While statement is similar to the *While* and *Do While* statements. However, in the ***While*** and ***Do While*** statements, the condition is tested at the beginning of the loop, before the body of the loop is executed whilst the ***Do ...Loop While*** tests the condition after the body of the loop has been executed. Therefore, in a ***Do ... Loop while*** statement, the body of the loop is always executed at least once. Again, when the condition becomes ***false***, the statement following the *Loop While* clause is executed. The syntax and activity diagram are as shown:



The following program illustrates the Do...Loop While structure:

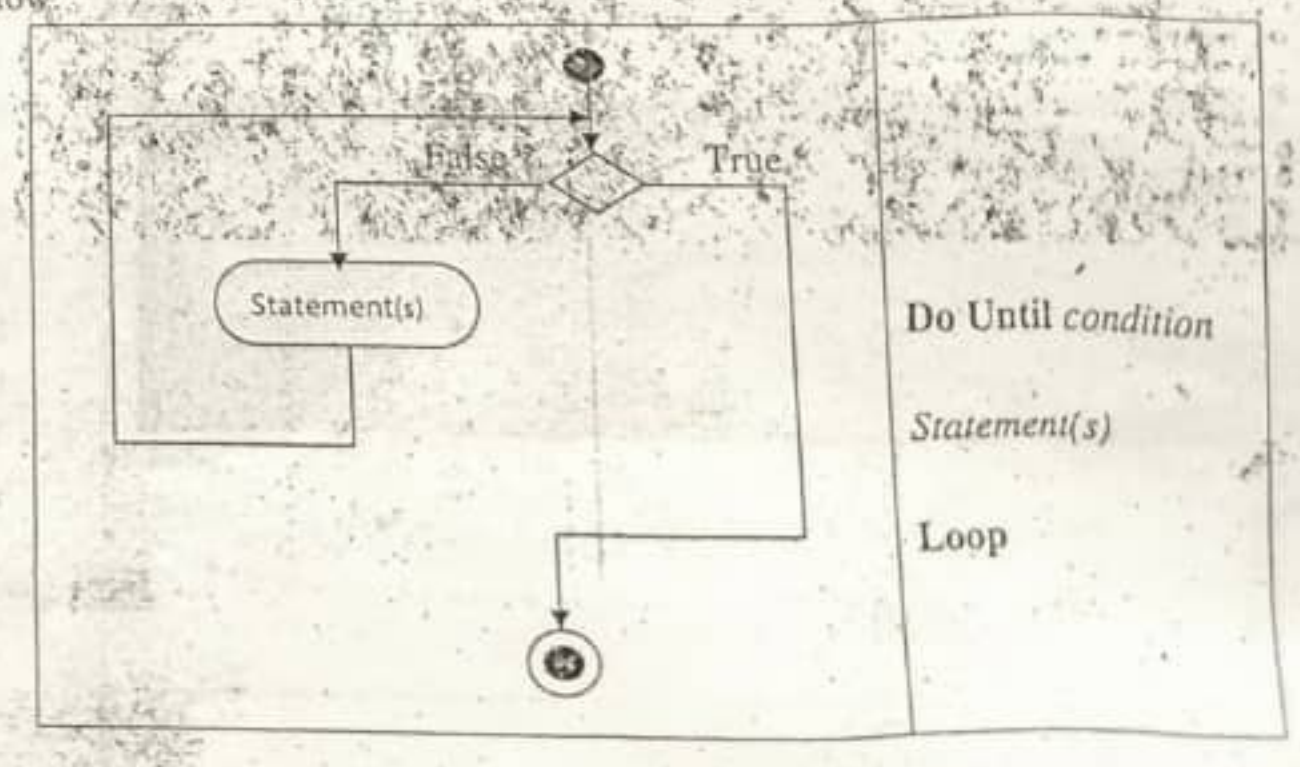
```
SubMain()
Dim sum AsInteger = 0
DimnumAsInteger = 1
Do
sum = sum + num
Console.WriteLine("value of number is: "&num)
num += 1
LoopWhilenum<= 20
Console.WriteLine(" Total value is:    "& sum)
Console.ReadLine()

EndSub
```


The program displays and accumulates the first 20 positive integers. Output of the program is as follows

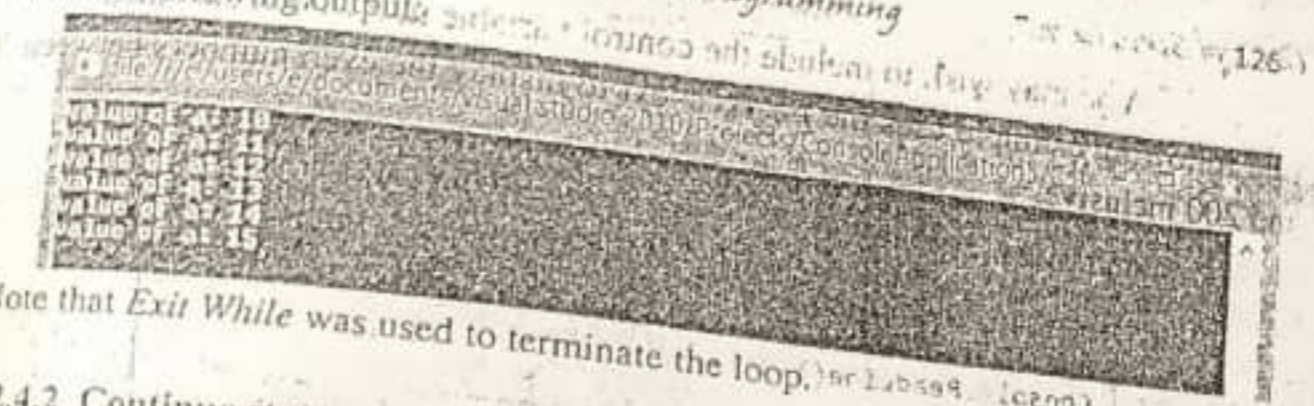


12.3.4 Do Until.. Loop

Again, unlike the While and Do While...Loop statements , the Do Until ...Loop statement tests for *falsity* for repetition to continue. This means that as long as the condition remains false, the body of the loop is executed. Execution terminates when the condition becomes true. Activity diagram and syntax of this structure are depicted below:



With the following output:



Note that *Exit While* was used to terminate the loop.

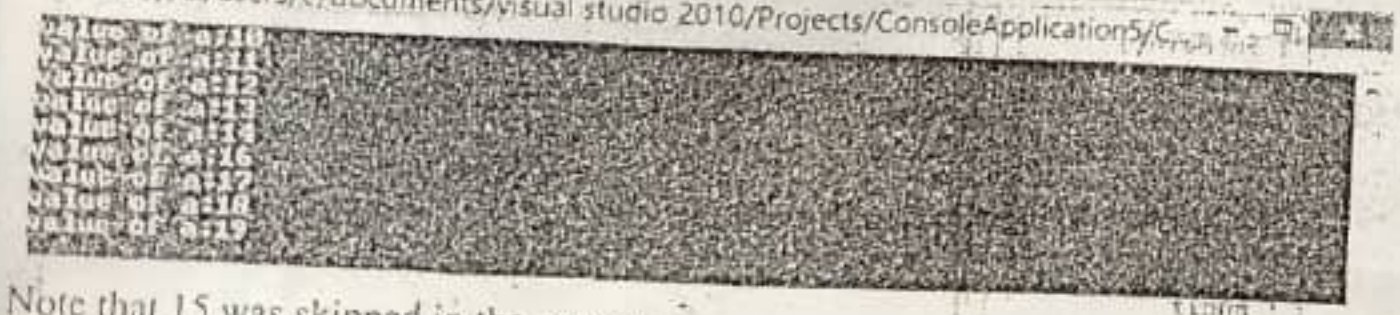
12.4.2 Continue statement: continue statement causes the loop to skip the remainder of its body and immediately retests its condition prior to reiterating. Control is then transferred to the next iteration of the loop. It has the syntax:

Continue [Do | For | While]

For example: Consider the following program:

```
Sub Main()
    Dim a As Integer = 10
    Do
        If (a = 15) Then
            a = a + 1
            Continue Do
        End If
        Console.WriteLine("value of a: {0}", a)
        a = a + 1
    LoopWhile (a < 20)
    Console.ReadLine()
End Sub
```

With the following output:



Note that 15 was skipped in the program.

Review Questions

1. What are the three classical control structures used in program writing
2. Give the syntax and activity diagram of the following structures (i.) If ...Then...Else (ii) Select case (iii) For...Next (iv) Do Until...Loop
3. What is the difference between the Do Loop ... While and the Do While... Loop
4. The control variable of the For ...Next structure must be of what data type
5. What happens when the step size in a For structure is 1
6. What are the uses of exit and continue statements. (b) Write a statement that exits a While loop
7. Write a program to generate the sequence 1, 11, 21, 31, 41 using (i) the While (ii) For...Next structures.
8. Write a program to generate and display the squares and cubes of numbers 1 to 20 using the Do Until ...Loop
9. Write a program to calculate and display the sum of the sequence: 1, 1/3, 1/5, 1/7, ..., 1/25 using the Do ....Loop While
10. Write a program to generate and display the factorial of the first 15 integers. (hint: factorial of n written as n! is given as n x (n-1) x (n-2) x (n-3) x ..., 1 (for n >= 1) and n! = 1 (for n = 0)
11. Identify and correct error(s) in the following program segments: (i) this should print the values 888 to 1000. Assume value is declared as an integer.

```
Value = 888
While value <= 1000
    Value = -1
End While
```

(ii) This should read an unspecified number of of positive values from the user and sum them. Assume that number and total are declared a integers

Chapter Thirteen

Arrays

13.0 Introduction

Consider a situation where you wish to store the scores made by 500 students in CSC 101. To use variables, it means that we would declare 500 variables in order store these scores. Of course this can be quite frustrating. To overcome this, an array can be used. An array is used to store a group of logically related elements, but it is often more useful to think of an array as a collection of variables of the same type. In an array, one element is stored after another in a linear way. For example: consider the scores of 10 students in CSC 102

Student 1 Score [0]	Student 2 Score[1]	Student 3 Score[2]	Student 4 Score[3]	Student 5 Score[4]	Student 6 Score[5]	Student 7 Score[6]	Student 8 Score[7]	Student 9 Score[8]	Student 10 Score[9]

By using an array, you can refer to these related values by the same name say Score, and use a number that is called an *index* or *subscript* to reference them. The individual values are called the ***elements*** of the array. They are contiguous from index 0 through the highest index value (9 in our example above). The size of an array is defined by its dimension. There are one dimensional array and multidimensional array. One dimensional array has *one subscript* and multidimensional array for example, *two dimensional* has *two subscripts.*

13.1 Declaring an array

To use an array, we should first declare it as we do for variables. Again, array names follow the same conventions that apply to variable names. The following syntax can be used to declare an array:

Form1: *Dim arrayname As datatype( ) = {initialvalues}*

Form 2: *Dim arrayname ( ) As dadatype = { initialvalues}*

Form 3: *Dim arrayname(uppersubscript) As datatype*

Where *Dim* stands for dimension, *arrayname* is the name of the array, *parenthesis* signifies that it is an array, *initialvalues* are initial values to be assigned to the elements of the array, *uppersubscript* is the maximum number of elements to be stored and *datatype* is the type of data that will be stored in it which could be any data

type such as *integer, char, Boolean* and *double*. For example, the following statement declares a one dimensional array called *faculty*

Dim *faculty* As string( ) = {" physc", "socsc", "mgt", "biosc", "law"}

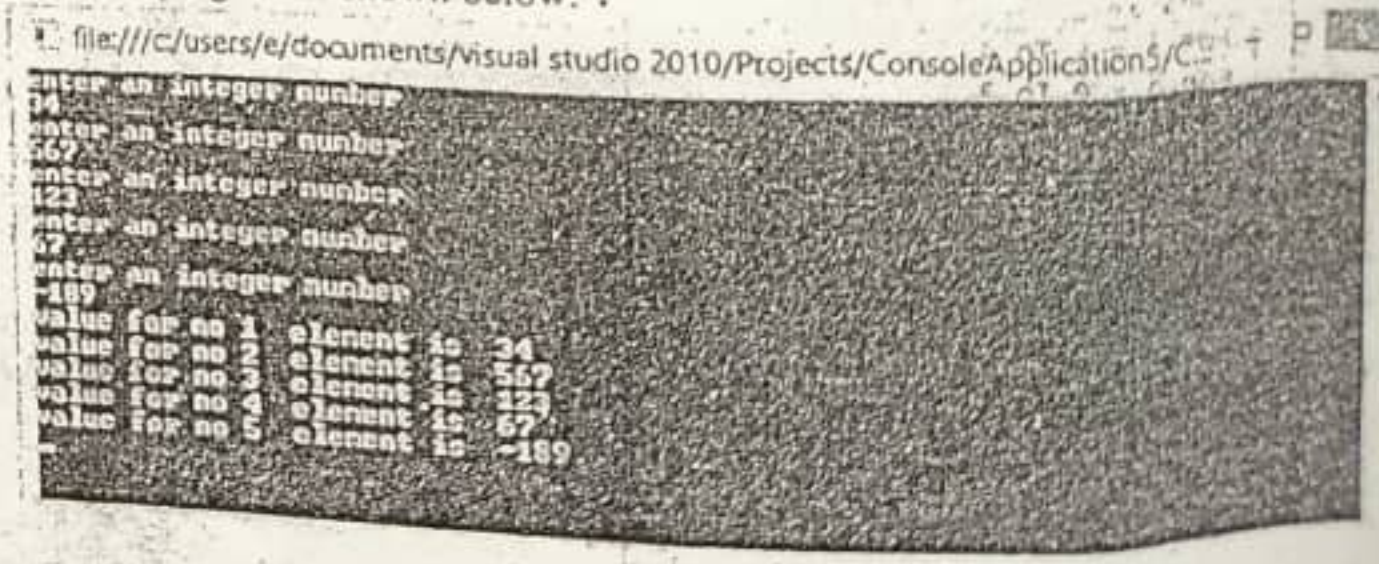
The array *faculty* contains 5 string elements with initial values of " *physc*", "*socsc*", "*mgt*", "*biosc*", and "*law*".

Let us now illustrate with a simple VB program that reads in 5 integer values to an array and writes them out.

```
1 Dimintegervalue(5) AsInteger
2 For k AsInteger = 1 To 5
3 Console.WriteLine("enter an integer number")
4 integervalue(k) = Console.ReadLine()
5 Next
6 For k AsInteger = 1 To 5
      Console.WriteLine("value for no "& k &"  element is  "&integervalue(k))
7 Next
8 Console.ReadLine()
```

Explanation:

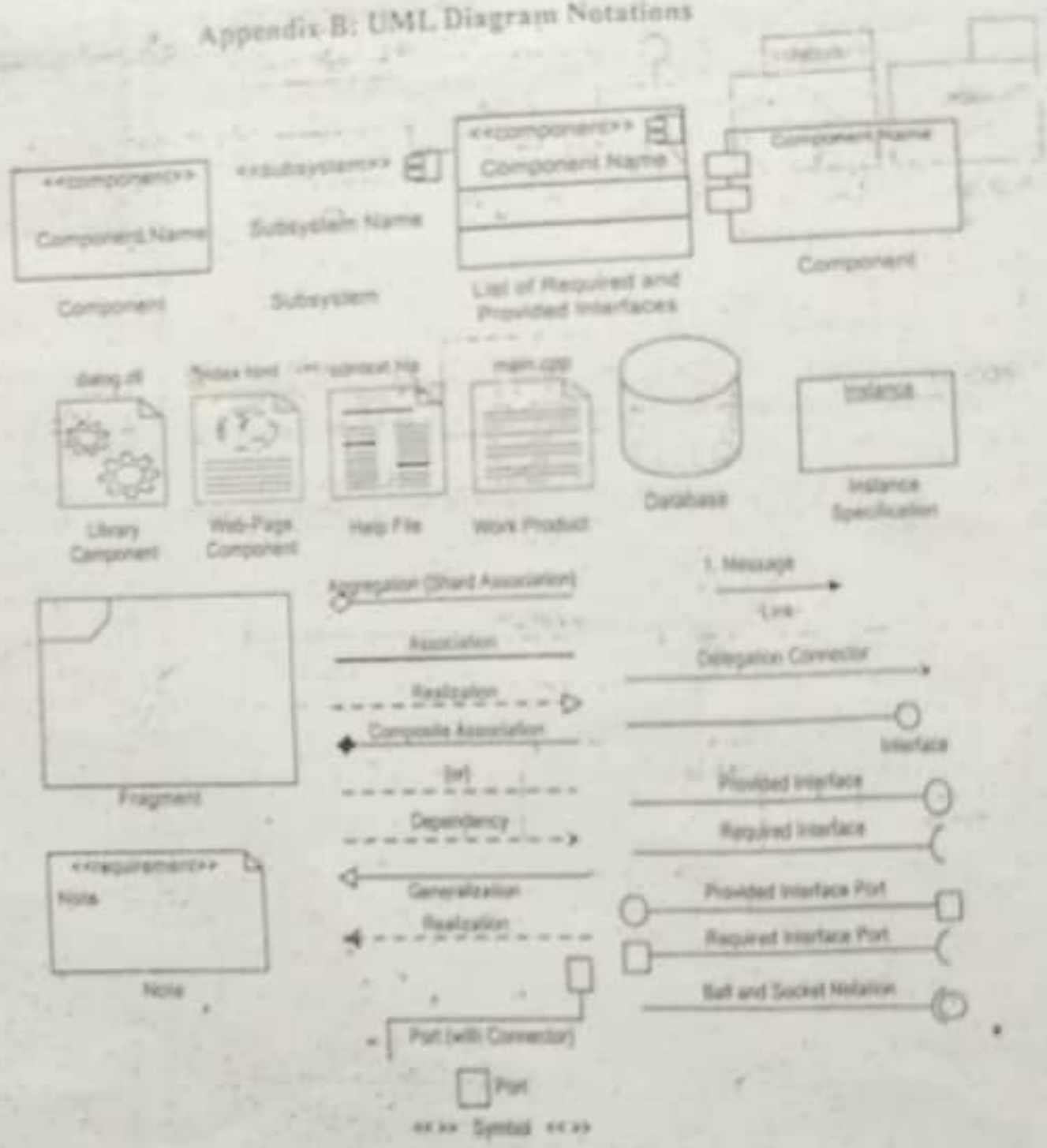
Line 1 declares an array called *integervalue* that will store maximum of 5 integ elements. Line 2 is a *For statement* that will enable us enter the 5 values (it w iterate 5 times). Line 3 will prompt the user to enter an integer value. Line 4 w assign the values entered by the user to the elements of *integervalue* . The seco *For ... statement* (line 6) is to display the values entered by the user. Sample output the above program is shown below: .



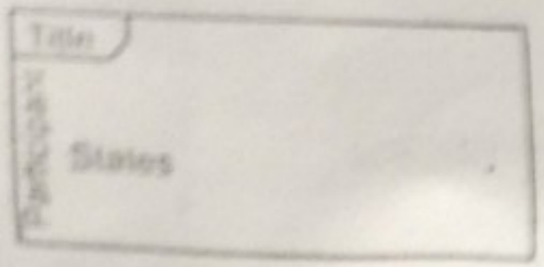
Appendix A: VB.NET Mathematical Functions

Mathematical Function	Description
Abs	Returns the absolute value of a specified number.
Acos	Returns the angle whose cosine is the specified number.
Asin	Returns the angle whose sine is the specified number.
Atan	Returns the angle whose tangent is the specified number.
Atan2	Returns the angle whose tangent is the quotient of two specified numbers.
BigMul	Returns the full product of two 32-bit numbers.
Ceiling	Returns the smallest integral value that's greater than or equal to the specified **Decimal** or **Double**.
Cos	Returns the cosine of the specified angle.
Cosh	Returns the hyperbolic cosine of the specified angle.
DivRem	Returns the quotient of two 32-bit or 64-bit signed integers, and also returns the remainder in an output parameter.
Exp	Returns *e* (the base of natural logarithms) raised to the specified power.
Floor	Returns the largest integer that's less than or equal to the specified **Decimal** or **Double** number.
IEEERemainder	Returns the remainder that results from the division of a specified number by another specified number.
Log	Returns the natural (base *e*) logarithm of a specified number or the logarithm of a specified number in a specified base.
Log10	Returns the base 10 logarithm of a specified number.
Max	Returns the larger of two numbers.
Min	Returns the smaller of two numbers.
Pow	Returns a specified number raised to the specified power.
Round	Returns a **Decimal** or **Double** value rounded to the nearest integral value or to a specified number of fractional digits.
Sign	Returns an **Integer** value indicating the sign of a number.
Sin	Returns the sine of the specified angle.
Sinh	Returns the hyperbolic sine of the specified angle.
Sqrt	Returns the square root of a specified number.
Tan	Returns the tangent of the specified angle.
Tanh	Returns the hyperbolic tangent of the specified angle.
Truncate	Calculates the integral part of a specified **Decimal** or **Double** number.

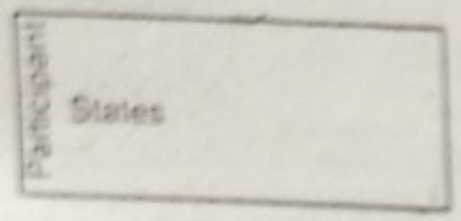
Appendix B: UML Diagram Notations



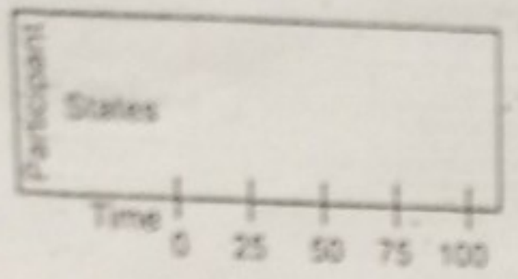
Component diagram notations



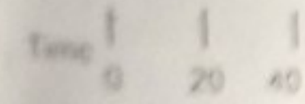
Top Lifeline



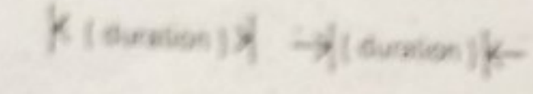
Lifeline

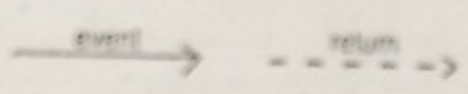


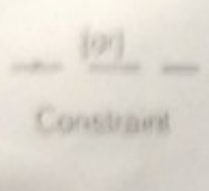
Bottom Lifeline



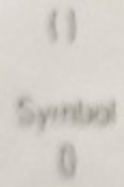
Time Intervals



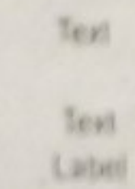




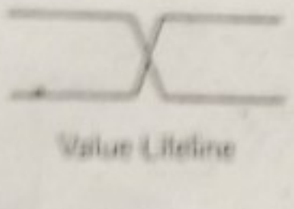
Constraint



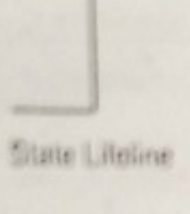
Symbol



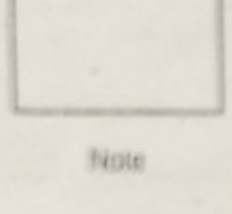
Text Label



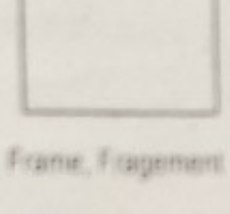
Value Lifeline



State Lifeline

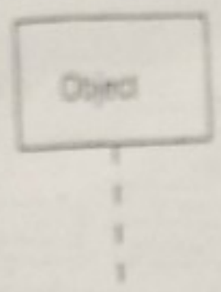


Note



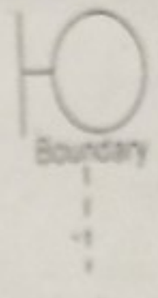
Frame, Fragement

Timing diagram notations

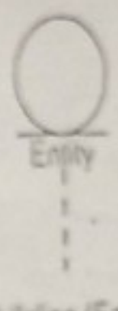


Lifeline

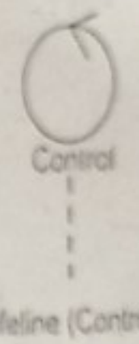
Line (Actor)



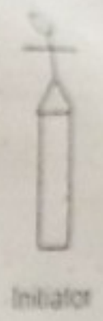
Lifeline (Boundary)



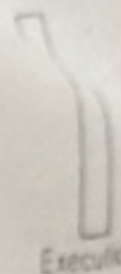
Lifeline (Entity)



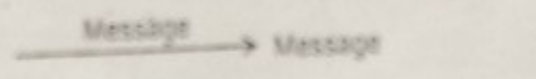
Lifeline (Control)

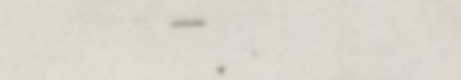


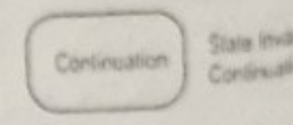
Initiator



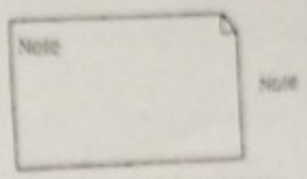
Execution Occurrence



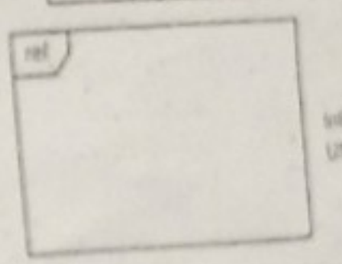




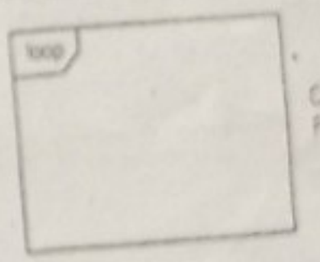
State Invariant/ Continuations



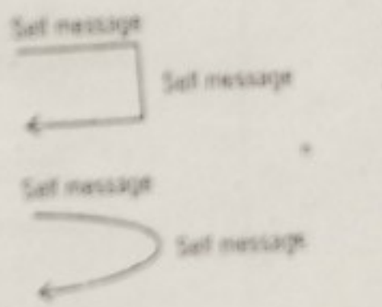
Note

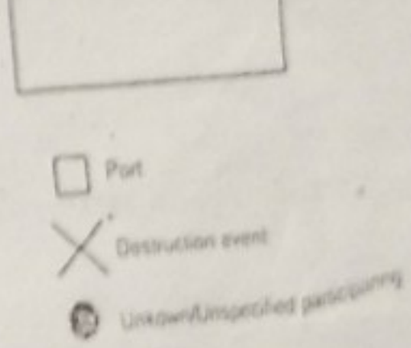


Interaction Use



Combined Fragment





Sequence Diagram notations