3055 BA SANGAM COLLEGE

PH: 6674003/9264117 E-mail: basangam@connect.com.fj

WORKSHEET 16

School: Ba Sangam College **Year / Level:** 11

Subject: Computer Studies **Name of Student**: ____________

Strand	3 – Programming
Sub strand	3.1 Programming
Content Learning Outcome	Describe generation of programming language

Five Generations of Programming Languages

Languages are described as occurring in "generations," from machine languages to natural languages.

There are five generations of programming languages. These are:

1. Machine Languages: The First Generation

We mentioned earlier that a byte is made up of bits, consisting of 0's and 1's. These 0's and 1's may correspond to electricity's being on or off in the computer. From this two-state system have been built coding schemes that allow us to construct letters, numbers, punctuation marks, and other special characters. Examples of these coding schemes, as we saw, are ASCII and EBCDIC.

Data represented in 0's and 1's is said to be written in machine language. To see how hard this is to understand, imagine if you had to code this:

111100100111001111010010000100000111000000101011

Machine languages also vary according to make of computer-another characteristic that makes them hard to work with.

2. Assembly Languages: The Second Generation

Assembly languages have a clear advantage over the 0's and 1's of machine language because they use abbreviations or mnemonics. These are easier for human beings to remember. The machine language code we gave above could be expressed in assembly language as

ADD 210 (8, 13), 028 (4, 7)

This is still pretty obscure, of course, and so assembly language is also considered low-level.

3. High-Level Procedural Languages: The Third Generation

People are able to understand languages that are more like their own (e.g., English) than machine languages or assembly languages. These more English-like programming languages are called "high-level" languages.

Procedural languages are programming languages with names like BASIC, Pascal, C, COBOL, and FORTRAN. They are called "procedural" because they are designed to express the logic - the procedures – that can solve general problems.

Compilers and Interpreters

For a procedural language to work on a computer, it must be translated into machine language so that the computer understands it. Depending on the language, this translation is performed by either a compiler or an interpreter.

A **compiler** converts the programmer's procedural language program, called the source code, into a machine language code, called the object code. This object code can then be saved and run later.

An **interpreter** converts the procedural language one statement at a time into machine code just before it is to be executed. No object code is saved. An example of a procedural language using an interpreter is the standard version of BASIC.

Activity

1. Differentiate between machine language and assembly language (2marks)

2. Why people prefer high level procedural language? (2marks

3. What is a compiler? (2marks

4. What is an interpreter? (2marks