1. **Data or Input**: representation of facts that can be manipulated and to which meaning can be assigned

2. **Data Processing**: the restructuring or manipulating of data to increase its usefulness and value

3. **Information or Output**: the meaning that is assigned to data; knowledge derived from data

Explanations:
- **Data**: two numbers addition sum
- **Data Processing**: set of quiz grades compute sum of grades and divide sum by count of grades average grade
- **Information**: grading CS 110A test
- **Information**: determining final grade for CS 110A student

4. **Basic Data Cycle**: input, processing, and output
   - **Input**: getting the data into the computer for the processing; keyboard
   - **Processing**: manipulating the data; processor
   - **Output**: getting the information out of the computer; monitor or printer

   **Hardware**
Importance of data cycle: recipe for most computer programs

Input Section:
- Input first number
- Input second number

Processing Section:
- Add two numbers

Output Section:
- Output sum
- End

5. **Softcopy Output**: output on the monitor (video screen)

6. **Hardcopy Output**: output on paper

7. **Computer Program**: a set of instructions that tell the computer what to do

8. **Computer System**: Electronic Data Processing (EDP) system capable of performing the following:
   - I/O (Input/Output) meaning input and/or output
   - Data Movement within the computer
   - Arithmetic Operations meaning addition, subtraction, multiplication, and division
   - Logical Operations meaning data comparisons

9. **Software**: the computer programs

10. **Hardware**: the physical equipment of a computer system
Hardware Components of a Computer System

11. **Peripheral Devices**: the I/O devices, that is, all input devices (keyboard), all output devices (printer), and secondary storage (disk drive)

12. **Central Processing Unit or CPU**: the hardware unit that executes or carries out the instructions of the program:
   a. 1 instruction at a time
   b. starting with the first (top) instruction
   c. proceeding in sequence top to bottom (normal sequence of execution)

The program must be located in main memory during program execution. Executing a program is sometimes called running a program.

13. **Control Unit**: supervises or controls the execution of program instructions
   a. fetches the instruction from main memory
   b. decodes the instruction
   c. sends the appropriate electronic signals to the hardware that will carry out the instruction

Note that the above 3-step cycle is repeated for each instruction in the program.
14. **Arithmetic/Logic Unit (ALU):** performs arithmetic and logical operations

15. **Main Memory or Primary Memory or RAM (Random Access Memory):**
a series of memory locations

<table>
<thead>
<tr>
<th>Addresses</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>30000</td>
<td>25</td>
</tr>
<tr>
<td>30001</td>
<td>5</td>
</tr>
<tr>
<td>30002</td>
<td>-1</td>
</tr>
<tr>
<td>30003</td>
<td>100</td>
</tr>
</tbody>
</table>

a. temporarily stores data and program instructions
   (Main memory is volatile.)
b. analogous to post office boxes, that is, each memory location has a numerical address (box number) and contents (letters, bills, magazines)
c. one location can store only one data item at a time
   (The new data replaces the old data.)
d. Data and program instructions are store in memory in a binary form

Humans use the decimal numbering system having a base of 10 and 10 decimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9

Computers use the binary numbering system having a base of 2 and 2 binary digits: 0 and 1

For example, 25 base 10 is stored as 11001 base 2.

16. **Bit:** Binary digIT

17. **Byte:** a group of bits, usually 1 byte is 8 bits

18. **Acronym:** a word formed from the initial letters of a group of words, as radar for RAdio Detecting And Ranging or snafu for Situation Normal All Fouled Up

19. **Input Devices:** enter data and program instructions into main memory
20. **Output Devices**: retrieve information from main memory

21. **Secondary Storage**: In order for a computer to execute a program, the program must be resident in main memory. In order for a computer to process data, the data must be resident in main memory. Therefore main memory must be used for the current program and its data. As main memory is finite and volatile, secondary storage (magnetic disk) saves both programs and data for future use. Main memory is temporary storage. Secondary storage is permanent storage.

22. **Machine Language**: the only programming language directly understood by a computer; it is coded in binary; it is a low-level language; programmer must be knowledgeable of hardware details; it is machine dependent; example instruction is 1100 0000 1110 0101 0011 1101 1111 0000

23. **Assembly Language**: a symbolic representation for machine language; machine language and assemble language are the two low-level programming languages; example instruction is ADD AX, BX

24. **High-Level Languages**: resemble English; eliminate the details that are needed in using machine language; can be used on most computers (Programs are said to be portable.)

- **Fortran** IBM Mathematical FORMula TRANslating System; 1955; the first high-level language(mid 50s); designed by group from IBM for scientific applications
- **COBOL** COMMON Business Oriented Language; 1960; for business applications
- **BASIC** Beginner's All-purpose Symbolic Instruction Code; 1965; easy-to-learn for students
- **Pascal** Blaise Pascal, French mathematician; 1971; structured programming
- **C** refinement of B which was a refinement of BCPL (Basic Combined Programming Language); 1972; by Dennis Ritchie of Bell Laboratories for programming the UNIX operating system
- **C++** early 1980s; by Bjarne Stroustrup of Bell Labs; added OOP (Object-Oriented Programming) features to C; C++ is a “better C”
25. **Compiler**: a computer program that translates a high-level programming language into the equivalent machine language. Compilers translate the entire high-level program (the source program) into a machine language program (the object program) before execution begins. Compilers are designed to translate one source language into one object language. The C++ source program is the input to the compiler, and the object program is the output from the compiler.

Source Program

```
... 
cin >> first_num;
cin >> second_num;
sum = first_num + second_num;
cout << sum;
```

Object Program

```
01100...
00000...
11000...
01010...
```

26. **Compilation**: the phase of running a program when the compiler translates the source code into object code; code means a set of program instructions

27. **Execution**: the phase of running a program when the CPU executes the instructions of the object program

28. **Operating System**: a complex set of programs that control the execution of all programs, allocate the computer’s resources (memory, processing time, I/O devices), and manage the computer’s interactions with users; examples are UNIX, DOS, and Windows.
Program Definition:

Input: 2 numeric values (15 and 5)

Processing: Compute the sum \((15 + 5)\),
            difference \((15 - 5)\),
            product \((15 \times 5)\), and
            quotient of the the two data values \((15 / 5)\).

Output:
Line 1: label for data values
Line 2: the two data values
Line 3: (blank)
Line 4: label for computed values
Line 5: sum of data values
Line 6: difference of data values
Line 7: product of data values
Line 8: quotient of data values
Line 9: (blank)
Line 10: closing message

The above is a description of the output, not the actual output.

Terms:
1. **label**: brief phrase that describes an output item

   Average grade is 85.4

2. **closing message**: brief message which clearly indicates that the program is done
Program Design:

3. **Program Logic or Algorithm:** the steps (program instructions), in their proper order, that are needed to solve the problem

4. **Flowchart:** graphical representation of program logic, consisting of symbols and flowlines. Each symbol represents a step of the logic and contains a brief description of that step.

```
Begin

Input First Number

Input Second Number

Compute Sum

Compute Difference

Compute Product

A

A

Compute Quotient

Output 2 Data Values

Output 4 Computed Values

Output Closing Message

End
```
The purpose of this exercise is to familiarize you with the procedures of running a C++ program in Batmale-301. The complete source program is shown below. The exact procedures for running the program are given in a separate handout.

The numbers down the left-hand side are reference numbers and not part of the C++ source code. You supply your own file name, name, and section number in lines 1, 2, and 3.

C++ is case-sensitive, that is, the case of a letter does matter. In general, lowercase is used more than uppercase.

```cpp
#include <iostream> // for cin, cout
using namespace std;

int main()
{
    int first_num, second_num; // Declare the variables
    int sum, difference, product, quotient;

    cout << "Enter the first number:\n";
    cin >> first_num;
    cout << "Enter the second number:\n";
    cin >> second_num;

    // Perform arithmetic processing
    sum = first_num + second_num;
    difference = first_num - second_num;
    product = first_num * second_num;
    quotient = first_num / second_num;

    // Print results
    cout << "Data values are:\n";
    cout << first_num << " and " << second_num << "\n";
    cout << "Computed values are:\n";
    cout << "Sum........... " << sum;
    cout << "Difference.... " << difference;
    cout << "Product........ " << product;
    cout << "Quotient...... " << quotient;
    cout << "End of Arithmetic Program\n";
    return 0;
}
```
The output screen produced by the program is shown below.

Note that the numbers to the left in italics refer to the program statements that generated the lines on the screen.

When you run the program, use 15 and 5 as your two data values.

Sample Run:
19 Enter the first number:
20 15<Enter>
21 Enter the second number:
22 5<Enter>
31 Data values are:
32 15 and 5
32
33 Computed values are:
34 Sum.......... 20
35 Difference.... 10
36 Product....... 75
37 Quotient...... 3
38
38 End of Arithmetic Program

prompt: brief message displayed on the monitor telling the user to enter some data
1. // is the comment indicator in C++; rest of line is a comment (no space between the two slashes)
   a. Comments include descriptive information in the program listing
   b. Read by another programmer who wants to understand the program
   c. Comments are not made available to the compiler
   d. Also known as Internal Documentation

2. 3 Forms of comments:
   a. **Single-line comment:**
      // comment
      Examples in pgm1.cpp are lines 1-3, 5-7, 18, 24, and 30.
   b. **Trailing comment:**
      C++ statement; // comment
      Examples in pgm1.cpp are lines 9 and 15.
   c. **Multiple-line comment:**
      // comment or /* comment
      // comment
      // comment
      comment
      comment */

3. Two kinds of comments:
   a. **Header comments** introduce the program (1 - 7).
   b. **Step comments** describe the major steps of the program logic (18, 24, 30).

4. You will learn commenting style as you learn programming.