```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    cout << "\tC++ Computer Programming" << endl;
    cout << setw (22) << "First year" << endl;
    cout << "Computer Tech. Engineering Department" << endl;
    cout << "  Madenat Alelem University College  \n";
    cout << "-------------------------------------" << endl;
    cout << setw(26) << "Assist. Lecturer\n";
    cout << setw(24) << "Haider Abdullah";

    return 0;
}
```

C++ Computer Programming

1. Algorithms

Merriam-Webster Online defines an **Algorithm** as "*a step-by-step procedure for solving a problem or accomplishing some end, especially by a computer*". Another definition is "*An algorithm is a set of well-defined instructions in sequence to solve a problem*". In mathematics, computing, linguistics, and related subjects, "*an algorithm is a sequence of instructions, often used for calculation and data processing*".

The algorithm concept is very old, ancient Babylonian mathematicians used arithmetic algorithms, such as a division. Greek mathematicians later used algorithms for finding prime numbers, and the Euclidean algorithm to find the greatest common divisor of two numbers. Arabic mathematicians such as al-Kindi in the 9th century used cryptographic algorithms for code-breaking, based on frequency analysis. The word algorithm itself is derived from the name of the mathematician Muhammad ibn Musa al-Khwarizmi (from Khwarazm) was Latinized as Algorithm.

Algorithms are commonly used to solve certain types of computational problems. We can often describe such a problem by specifying a relationship between input and output.

A sequence of English statements can also be an algorithm. An algorithm can be expressed in English like language, called pseudocode, in a programming language or the form of a flowchart.

It is difficult to write direct programs in any language, just like you cannot start constructing a building without the design of a building. For constructing a building, you need a design of building, similarly for writing a large or good program you need an algorithm.

1.2 Algorithm Specifications

- **Input**: Zero or more quantities are externally supplied.
- **Output**: At least one quantity is produced.
- **Definiteness**: Each instruction must be unambiguous (Unique meaning).
- **Finiteness**: An algorithm terminates in a finite number of steps.
- **Effectiveness**: Every instruction must be basic enough to be carried out then, which means not so complex.

1.3 Pseudocode

Pseudocode is a compact and informal high-level description of an algorithm intended for humans rather than machines. It's simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming languages and thus can't be compiled or

interpreted by the computer. The main goal of pseudocode is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer. There are lots of formats used for writing pseudocodes and most of them borrow some of the structures from popular programming languages such as C, Lisp, FORTRAN, etc. Also, natural language is used in Pseudocode when presenting details that are not important.

Most of the algorithms are presented using pseudocode since they can be read and understood using programmers who are familiar with different programming languages. Some programming constructs used for Pseudo Code – *READ, PRINT, SET, INITIALISE, INCREMENT, IF-THEN – ENDIF, IF – THEN – ELSE – ENDIF, REPEAT – UNTIL, DO – WHILE* etc.

1.4 Algorithm and Pseudocode

- An algorithm is a well-defined sequence of steps that provides a solution for a given problem, while pseudocode is one of the methods that can be used to represent an algorithm.
- Algorithms can be written in natural language, while pseudocode is written in a format that is closely related to high-level programming language structures.
- Pseudocode does not use specific programming language syntax and therefore could be understood by programmers who are familiar with different programming languages.
- Transforming an algorithm presented in pseudocode to programming code could be much easier than converting an algorithm written in natural language.

Algorithm:

```
Input a set of 4 marks
Calculate their average by summing and dividing by 4
if the average is below 50
    Print "Fail"
else
    Print "Pass"
```

pseudocode

```
Step 1: Input M1, M2, M3, M4
Step 2: avg (M1+M2+M3+M4)/4
Step 3: if (avg < 50) then
    Print "Fail"
else
    Print "Pass"
endif
```

1.5 Problem-solving

1. Understanding the problem.
2. Identify the program input and output.
3. Design the algorithm (solution).
4. Writing a code in any programming language to apply the algorithm.

Problem: TV remote control doesn't work
Algorithm: To replace the battery
Step1: open remote-control cover
Step2: replace the old battery
Step3: Close the cover
Step4: TV power on
Step5: Check if the remote-control works

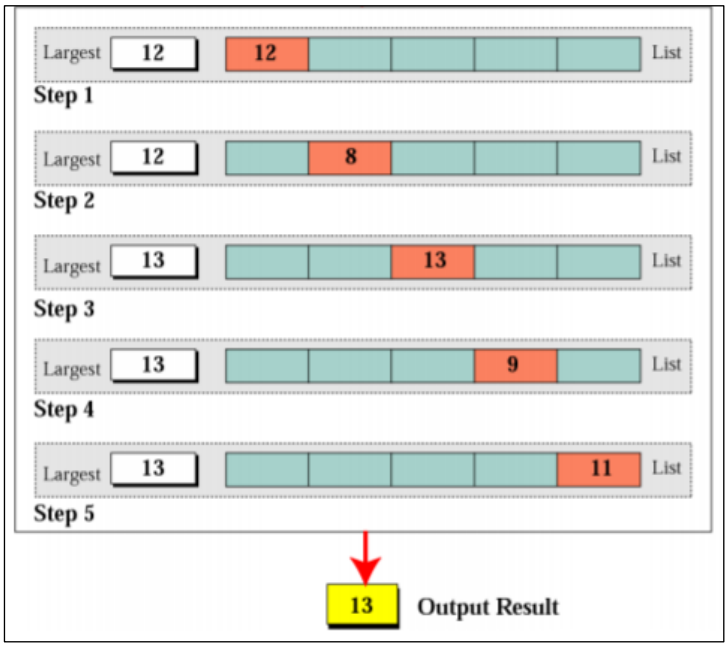
Problem: To print the numbers from 1 to 50
Pseudocode: PrintNumbers1-50
1. FOR number = 1 to 50
2. Print number
3. END FOR

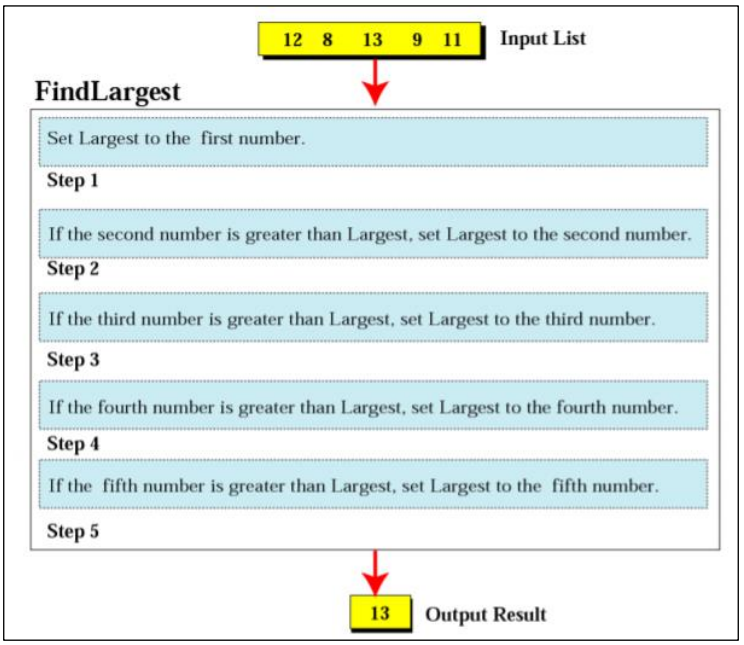
1.6 Algorithm and Program

- A program is an implementation of an algorithm to be run on a specific computer and operating system.
- An algorithm is more abstract – it does not deal with machine-specific details – think of it as a method to solve a problem.

What is a good algorithm?
Efficient algorithms are good, we generally measure the efficiency of an algorithm based on:
1. **Time**: the algorithm should take minimum time to execute.
2. **Space**: the algorithm should use less memory.

Example1: Defining actions of an algorithm to find the largest number among a group of numbers.





Example2: Find the area of a circle of radius r.

Algorithm: AreaofCircle
Step1: start
Step2: read the radius r of the circle
Step3: area = PI*r^2
Step4: Print Area
Step5: end

Example3:

Algorithm: AverageOfTwo

1. start
2. Input: Two numbers
3. Add the two numbers
4. Divide the result by 2
5. Return the result by step 3
6. End

Example4: Pass/Fail pseudocode

Pseudocode: Pass/FailGrade
Input: number
if (number >= 40)
then
print "pass"
else

```
            print "fail"
        End if
            Return grade
End
```

Example5:

Algorithm: FindtheGrade

```
Input: One number
        if (the number is between 80 and 100)
then
        Set the grade to "A"
End if
        (the number is between 70 and 79)
then
        Set the grade to "B"
End if
        (the number is between 60 and 69)
then
        Set the grade to "C"
End if
        (the number is between 50 and 59)
then
        Set the grade to "D"
End if
        (the number is less than 50)
then
        Set the grade to "F"
End if
        Return the grade
End
```

Example6: Largest number algorithm

Pseudocode: FindLargest()

```
Input: 1000 positive integers
    Largest = 0
    Counter = 0
    while (Counter < 1000)
    if (integer > Largest)
```

```
            then
        Set Largest = integer
    End if
        increment Counter = Counter + 1
    End while
        Return Largest
    End
```

Example7: Write a Pseudocode to find factorial for a number.

Pseudocode: FindFactorial

1. Read number
2. Fact = 1
3. i = 1
4. WHILE i <= number
5. Fact=fact*i
6. i=i+1
7. ENDWHILE
8. WRITE Fact

H.W1: Design an algorithm that generates odd numbers between 1000 and 2000 and print them and their total sum in standard output.

H.W2: Design an algorithm to calculate the following formula:

$Y = 1 + 1/3 + 1/5\ldots + 1/x$

H.W3: Write an algorithm to calculate even numbers between 0 and 99.

H.W4: Read a number and check if it's a prime number (use Pseudocode).

H.W5: Write an algorithm and pseudocode to ask a user to enter a number and check the following: If the number is between 0 and 10, write the word "Black". If the number is between 10 and 20, write the word "yellow". if the number is between 20 and 30, write the word "Blue". If it is any other number, write "Invalid colour option".