Part 1 - Introduction to Computers

1) Introduction to Computers

# Hardware
# Software

2) Operating System

3) Visual Basic Environment

# IDE
# Compiler
# Debugger
What is a Computer?

# An Electronic digital device that can store, and process data. The computer accomplishes this task by using a **program**.

What is a Program?

# A plan to achieve a solution to a problem.

# A set of sequential instructions, which cause a computer to perform a particular operation or task.

Hardware vs. Software

# **Hardware** is the actual circuitry and physical equipment of a computer system. **Hardware** is Tangible.

# **Software** is a sequence of instructions that are stored either magnetically or electronically. **Software** is non-tangible.
Hardware:

# CPU (The Brain) (AKA Microprocessor)
# BUS (Central Nervous System) (Series of parallel wires, which allow various component to communicate with each other.)
# Memory (Storage) (ROM, RAM, Byte, Bit)
# Secondary Storage (Disks, Tapes) (Floppy, Hard disk)
# Input Devices
# Output Devices

What are Some Basic Computer Operations?

# Add, Subtract, Multiply and Divide two memory locations
# Read from, and write to I/O devices (Keyboard, Screen, Printer)
# Read from, and write to Storage devices (Disk, tape, etc..)
# Move the contents of one memory location to another
# Perform Logical operations (Test for a condition and act on it)

    if A > B then
        go home
    else
        work some more

How is information represented inside the Computer?

# Binary Format (Why?)

# ASCII Code (American Standard Code for Information Interchange)

# UNICODE
Components of a Computer System

Input Devices

# Keyboard, Mouse, Light pen, Track ball, Joy stick

Processor Unit

# Memory (ROM and RAM)
   # ROM (Read only Memory) (Permanent)
   # RAM (Random Access Memory) (Volatile)
   # Byte (Unit of memory)
   # 1 Byte = 8 Bits, (Bit = Binary Digit. 0,1)

# Arithmetic Logic Unit (ALU)
   # Calculations and decision making

# Control Unit (CU)
   # Control the flow and maintains the order of operations. (Moves data from memory to CPU and Back)
Secondary Storage Devices

# Used to supplement the limited storage capacity of primary memory.

# **Floppy Disks:** (Removable)
# Made out of Mylar Plastic.
# Random Access Device
# Information is magnetically stored

# **Hard Disks:** (Usually Fixed)
# Made out of metal.
# Capacity ranges from 30 or 40 Meg to Many Giga bytes.
# Costs about .15c per megabyte.
# Random Access Device
# Faster than Floppy disk or tapes
# Information is magnetically stored

# **Optical Disks:**
# Similar to Audio CD’s
# Very high capacity (Approx. 600 Megabytes)
# Random Access Device
# Uses laser light to read the disk.

# **Flash Memory:**
# Non-volatile EEPROM (Electrically Erasable Programmable Read-Only Memory)

Output Devices

# Display Screen (monochrome, CGA, EGA, VGA, SVGA)
# Printers (Dot Matrix, Letter Quality, Laser, Ink Jet, Thermal)

Communication Devices
# Modem

# Network Cards
Operating Systems

# An organize collection of software programs that control the overall operation of the computer.

# A set of programs which act as the interface between the user and the computer hardware.

# Examples: (Windows XP, UNIX, Linux, etc.)
Programming Languages

# Machine Language

# Low level languages (Assembly Language)

# High Level Languages (Basic, Pascal, C, C++, COBOL)

# 4GL's (AKA Rapid Application Development or RAD)
Machine Language:

# 0's and 1's
# Instructions are written in 0's and 1's
# Example:
# 0110 10001 10100 10110
#     \ \ \ \    \
ADD X Y Z

Low level Language:

# Low level or Assembly Language provides a one to one mapping between a symbol (name) and computer instructions and memory locations.
# Tightly coupled to the machine language
# Example:
# ADD X, Y, Z

High level language:

# Closer to English
# Basic, Fortran, C, C++, Pascal, Cobol, etc.
# Easier to write and debug programs.
# Example:
# Z = X + Y
# If (Z >= 100)
    cout << "Z is big" << endl;
# Any high level language must eventually be translated to machine before the computer can understand it.
What is a Compiler?

# A program that translates a high level language into machine language.

Steps in Writing, Compiling and Running a program:

Editor
Write Program

Compiler
Compile the Program

*.cpp

*.EXE

Run the .EXE file
Software Development Life Cycle

1) Requirement specification

# Problem Definition

2) Analysis

# Input format
# Output format
# Requirements (formulas, facts, figures, etc..)
# Constraints (limits, etc..)
# Identify possible solutions

3) Design

# Develop a list of steps (an Algorithm) to solve the problem.
# An algorithm is often written in a generic language called pseudocode.
# Desk check your algorithm.

4) Implementation

# Convert the algorithm developed in the Design phase into a desired programming language (i.e. VB).

5) Testing

# Verify the correctness of your program. (component testing, and overall testing.

6) Maintenance

# Add, modify, and maintain the system.
Problem Solving Techniques

1) Ask questions to fully understand the problem
   # What is my data?
   # What does the data look like?
   # How much data is there?
   # How will I know when I have processed all the data?
   # What should the output look like?
   # How many times is the process going to be repeated?
   # What special error conditions might arise?

2) Identify patterns
   # Reuse previous solutions, do not reinvent the wheel

3) Solve by analogy
   # Think of similar problems and how you solved them.

4) Means end analysis
   # Given a set of input data, how do we reach the desired output results, provided a set of tools at our disposal.

5) Divide and conquer
   # Divide the problem into a series of smaller, more manageable problems.

6) Merging Solution
   # Some problems can be viewed as a combination of 2 or more existing problems. (merge the existing solutions to get the new solution!)

7) Start by starting
   # Start by rewriting the problem in your own words. Try explaining the problem to your friend.

8) Algorithmic Problem Solving
   # Develop a step by step solution to a problem and then refine it.
(step wise refinement)
Sample Problems for class discussion:

1) Calculating the area and circumference of a circle.

2) Calculating Student grades

3) Finding the Smallest and Largest number
Writing your first Algorithm

Problem: Given the radius of a circle, compute and display the **Area** and the **Circumference**.

Analysis:

# Determine the Input and Output of the program:
- Radius of a circle

# Determine the formulas, fact, etc. needed:
- Area = PI * (Radius)^2
- Circumference = 2 * PI * Radius
- PI = 3.14159

Design:

# Develop a list of steps to solve the problem (An algorithm)

1) read the radius
2) compute the Area
   2.1) Area = PI * (Radius)^2
3) compute the circumference
   3.1) Circumference = 2 * PI * Radius
4) print the area and circumference

Implementation:

```vbnet
Module Module1
    Sub Main()
        Dim radius, area, circumference As Double
        Const PI = 3.14159

        Console.Write("Please enter the radius of the circle? ")
radius = Console.ReadLine()
area = PI * radius * radius
circumference = 2 * PI * radius
Console.WriteLine("Area = " & area)
Console.WriteLine("Circumference = " & circumference)

        Console.ReadLine()      'just to pause the program.
    End Sub

End Module
```

A201 - A505 Object Oriented Programming (VB . Net)  Copyright 2007  15
End Module
Calculating Student Grades

PART I

Write a program which accepts student names and test scores (between 1 and 100) as input. The program should then produce an output similar to the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Score</th>
<th>Letter Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>93</td>
<td>A</td>
</tr>
<tr>
<td>Jack</td>
<td>75</td>
<td>C</td>
</tr>
</tbody>
</table>

If the user enters a negative number your program must terminate the program.

The Letter Grades are calculated as follows:

\[
\begin{align*}
&>=90 \quad A \\
&>=80 \quad B \\
&>=70 \quad C \\
&>=60 \quad D \\
&< 59 \quad F
\end{align*}
\]

Part II

Calculate and print the average for the class:

Class Average: 84 = B

Visual Basic language constructs needed for this problem:

Arithmetic Operators, if-else, loops (while or for), input, output.
Calculating Student Grades

Problem Definition:
  #  Do you understand the problem? If so write it in your own words

Analysis:
  #  Determine the Input and Output of the program
  #  Determine the formulas, fact, etc.

Design:
  #  Develop a list of steps to solve the problem (An Algorithm)

Implementation:
Finding the Smallest and Largest Integer Value Entered by the User

Problem Definition:
# Do you understand the problem? if so write it in your own words

Analysis:
# Determine the Input and Output of the program
# Determine the formulas, fact, etc.

Design:
# Develop a list of steps to solve the problem (An Algorithm)
Implementation:

Module Module1

Sub Main()
    Dim HowMany, Smallest, Largest, Number As Integer
    Console.WriteLine("Enter a series of positive integers. (0=stop)")
    Number = Console.ReadLine()
    HowMany = 0
    Smallest = Number
    Largest = Number
    Do While Number <> 0
        HowMany = HowMany + 1
        If Number > 0 Then
            If Number < Smallest Then
                Smallest = Number
            ElseIf Number > Largest Then
                Largest = Number
            End If
        Else
            Console.WriteLine("No negative numbers")
        End If
        Number = Console.ReadLine()
    Loop
    Console.Write("Altogether, you entered ")
    Console.Write(HowMany)
    Console.WriteLine(" numbers.")
    Console.Write("The smallest was ")
    Console.WriteLine(Smallest)
    Console.Write("The largest was ")
    Console.WriteLine(Largest)

    Console.ReadLine()

End

End Sub

End Module