**SECTION A: Understanding Exception Handling**

1. Understand what an exception is.
2. Define error handling.
3. Explain different types of errors.
4. List the advantages of exception handling.

* Exception: an event caused by an error and interrupts program execution.

Ex: when a traffic accident occurs, medical people and police handle this exception.

TYPES OF ERRORS:

* Run-time: occurs during program execution. EX: not enough memory, wrong user input.
* Compile-time: occurs during compilation. Caused by errors in creating/editing code. EX: syntax, code error.
* Logic: may not cause program to crash or terminate and does not produce error message. Results in undesired/unintended output.

**SECTION B: Understanding Pre-Defined Exceptions**

1. Understand the types of exceptions in VB.NET.
2. Determine pre-defined exceptions and how to use them in coding.

* Pre-defined: all system exceptions that occur during run-time and compile-time. Ex: out of memory, arithmetic exception.
* User-defined: custom error message created by programmer’s logic make user fix their input. Ex: error message for invalid passwords.

PRE-DEFINED:

* Out of Memory: occurs when there is too much data for memory to hold.
* Stack overflow: occurs when user overflows a stack and stops program’s response. Create error message to user using stack overflow exception class.

Ex: can be caused by division by 0.

* Null reference: occurs when object (such as a name) or variable are assigned with no value.
* Index out of range: when the specific value does not exist in an array.

Ex: when asking for value from a(10] in the array called a that has values from 0 to 9.

* Arithmetic: handles all mathematical operation errors.
* Is the base class for DivideByZeroException, NotFiniteNumberException, and OverflowException.

**SECTION C: User-defined Exceptions:**

1. Define user-defined exceptions.
2. Write a program with a user-defined exception.

User-defined Exceptions Keywords:

Try: code in this block is monitored for an exception.

Catch: process to gather possible reasons for error from the try block in a variable (ex).

Throw: code tells the user the error.

Finally: executes when try block is finished, or after the throw block. This runs regardless if try block finds an exception or not. Last chance to write code needed to be executed before exception jumps out of control.

Ex: if the try block establishes connection to a database via a connection object, the finally block must clean up this unwanted object to cut off connection after job is done.

\*\*\*Shared: same as static in Java. Sharing member of class or structure makes it available to every instance in that class or structure, and to outside classes/structures.

\*\*\* Nonshared: each instance keeps its own copy.

Compile-Time Error Example:

