\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC253 ADVANCED C# ProGRAMMING**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 09 **EXCEPTION HANDLING**

# Objectives

In this lab assignment, students will learn:

- How to write robust programs

- How to detect an exception

- How to handle an exception

- How to write an exception class

# Goals

In this lab assignment, students will demonstrate the abilities to:

- Write robust programs

- Detect an exception

- Handle an exception

- Write an exception class

# Description

Create a C# console application for each question. When you create a new C# project, Visual Studio creates a folder to hold every file and sub-folder for your project. You need to zip this folder and submit the zip file to Blackboard.

Factorial of a non-negative integer n (i.e. n!) is defined as follows:

n! = 1 if n = 0

n! = 1 \* 2 \* 3 \* 4 \*… \* (n-1) \* n if n >= 1

For example,

4! = 1 \* 2 \* 3 \* 4 = 24

Write a C# program to calculate factorial. The user enters a positive integer n. The program will calculate and display n!. Your program should be able to handle three types of exceptions.

FormatException: This happens when the user enters something that cannot be converted to int type value. When this type of exception happens, display the error message Invalid number format in the console window and ask the user to try again. Example:

Enter a non-negative integer: 4.5

Invalid number format. Please try again.

NegativeNumberException: This happens when the user tries to find the factorial of a negative integer. The .NET exception hierarchy does not have this exception. You must create it yourself. You can use the code in Figure 13.6 of the textbook. When this type of exception happens, display the error message “Factorial of negative integer not permitted” in the console window and ask the user to try again. Example:

Enter a non-negative integer: -4

Factorial of negative integer not permitted. Please try again.

OverflowException: This happens when the factorial is too large. The value of factorial increases rapidly. An overflow happens for n > 12. To detect overflow, you must use the checked function. Example:

int f = 0;

f = checked (a \* b);

If the result of a \* b is larger than what f can store, an overflow exception will be thrown.

Display the error message Input value too large in the console window and ask the user to try again if an overflow exception happens. Example:

Enter a non-negative integer: 40

Input value too large. Please try again.

The following figure shows various types of overflow caught.

Enter a non-negative integer: -5

Factorial of negative integer not permitted. Please try again.

Enter a non-negative integer: 2.4

Invalid number format. Please try again.

Enter a non-negative integer: 20

Input value too large. Please try again.

Enter a non-negative integer: 10

10! = 3628800

Press any key to continue . . .

# Grading rubric

Catch and handle format exception [25 pts]

Catch and handle negative integer exception [25 pts]

Catch and handle overflow exception [25 pts]

Calculating factorial [25 pts]