CL-1004
Object Oriented Programming

Lab No 12

Objectives:

- Exception Handling

Note: Carefully read the following instructions (*Each instruction contains a weightage*)

1. Use proper **font family (Calibri or Times New Roman)** and **font size** of the **title (16 points), heading (14 points), subheading (12 points),** and **normal text (10 points)**.
2. First think about the problem statement and then write/draw your logic on paper.
3. **Microsoft Visual Studio** should be used to make c++ programs. Programs made with any other software would not be accepted.
4. For each task in the manual create a new C++ program with the naming convention as follows:
   TASK-NO
5. **Mention what is happening in each line of code using comments**.
6. Write all codes one by one with proper numbering and also paste screen shot of each problem using the **snipping tool** (default screen capture software in windows) on **Microsoft word file.**
7. Please submit your file with this naming convention **ROLLNO_SECTION_GROUPNO_LABNO**.
8. Do not copy from any source otherwise, you will be penalized with zero marks.
9. Submit your lab on **Google Classroom**.

Problem 1: Exception Handling

a. Write a simple program using function that throws an exception if divide by zero occur and catch the exception in main.

b. Write a program to throw and catch the following type of exceptions:

   1. Integer i.e. throw 1
   2. Float
   3. String i.e. throw “abc”
   4. Character

Write one try block and appropriate specific catch block/s.

Problem 2: Exception Handling

Write a program that lets the user perform arithmetic operations on fractions. Fractions are of the form a/b, in which a and b are integers and b! = 0. Your program must be menu driven, allowing the user to select the operation (+, -, *, or /) and input the numerator and denominator of each fraction.

Furthermore, your program must consist of at least the following functions:

a. **Function menu:** This function informs the user about the program’s purpose, explains how to enter data, and allows the user to select the operation.

b. **Function addFractions:** This function takes as input four integers representing the numerators and denominators of two fractions, adds the fractions, and returns the numerator and denominator of the result.

c. **Function subtractFractions**: This function takes as input four integers representing the numerators and denominators of two fractions, subtracts the fractions, and returns the numerator and denominator of the result**Function multiplyFractions:** This function takes as input four integers representing the numerators and denominators of two fractions, multiplies the fractions, and returns the numerators and denominators of the result.

d. **Function divideFractions:** This function takes as input four integers representing the numerators and denominators of two fractions, divides the fractions, and returns the numerator and denominator of the result.

Code the above problem such that your program handles Exceptions such as division by zero and invalid input.

Problem 3: Exception Handling

Consider the following C++ code:

```
int lowerLimit;
try
{
cout << "Entering the try block." << endl;
if (lowerLimit < 100)
throw exception("Lower limit violation.");
cout << "Exiting the try block." << endl;
}
catch (exception eObj)
{
cout << "Exception: " << eObj.what() << endl;
}
cout << "After the catch block" << endl;
```

Explain what if the output:

a. The value of lowerLimit is 50?

b. The value of lowerLimit is 150?

Problem 4: Exception Handling

Write a program that prompts the user to enter a length in feet and inches and outputs the equivalent length in centimeters. If the user enters a negative number or a non-digit number, throw and handle an appropriate exception and prompt the user to enter another set of numbers.

Problem 5: Exception Handling

Write a program that prompts the user to enter time in 12-hour notation. The program then outputs the time in 24-hour notation. Your program must contain three exception **classes**: **invalidHr**, **invalidMin**, and **invalidSec**. If the user enters an invalid value for hours, then the program should throw and catch an **invalidHr** object. Similar conventions for the invalid values of minutes and seconds.

Problem 6: Exception Handling

Write a program that prompts the user to enter a person's date of birth in numeric form such as 8-27-1980. The program then outputs the date of birth in the form: August 27, 1980. Your program must contain at least two exception **classes**: **invalidDay** and **invalidMonth**. If the user enters an invalid value for day, then the program should throw and catch an **invalidDay** object. Similar conventions for the invalid values of month and year.