Programming Laboratory-I

Assignment No-6

(Basics of object oriented programming) Due date: 13/10/23

1. EXCEPTION HANDLING

Write a program to perform arithmetic operations on any given two numbers which throws an exception if the dividend is zero or does not contain an operator.
Use of multiple try-catch statements is mandatory to perform this assignment.
Provide specific ERROR names for the errors occurring during the exception handling

2. EXCEPTION HANDLING

Write a program to accept an email address and throw an exception if the given string does not contain '@' symbol in it
To perform this assignment, use of exception and its keywords is mandatory.
Use of multiple try-catch statements is mandatory to perform this assignment.
Provide specific ERROR names for the errors occurring during the exception handling

3. TEMPLATE

Write a program for a function template for finding the minimum value contained in an array.

Expected Output:
The minimum value of array (6,8,3,1,9) is = 1
The minimum value of array (1.3,1.5,0.5,0.200,2.5) is = 0.200

4. EXCEPTION HANDLING

Design and implement a C++ program for a basic banking system with the following features:

1. Create a `BankAccount` class that represents a bank account. Each account should have the following attributes:
   - Account holder's name
   - Account number (a unique identifier)
   - Current balance

2. Implement exception classes for the following scenarios:
    - `NegativeBalanceException`: This exception should be thrown when a user attempts to withdraw more money than their account balance, resulting in a negative balance.
    - `InvalidAmountException`: This exception should be thrown when a user attempts to deposit or withdraw a negative amount or zero.
3. Your program should provide a menu-driven interface with the following options:
    - Create a new account.
    - Deposit money into an account.
    - Withdraw money from an account.
    - Check the account balance.
    - Quit the program.
4. Implement exception handling for the following situations:
    - Catch and handle exceptions related to negative balances and invalid amounts, ensuring that the program does not crash.
    - Display appropriate error messages when exceptions are caught.
    - Ensure that account balances are not modified when exceptions occur.
5. Ensure that the account numbers are unique for each account created.
6. Use appropriate object-oriented programming principles to design your classes and functions.

TEMPLATE

Design and implement a C++ program that provides generic data structures for managing students and courses in a university system. Your program should include the following components:

1. **Generic Linked List Template**:

   Create a generic linked list template class called `LinkedList` that can store data of any data type. It should support basic operations such as insertion, deletion, and traversal. Demonstrate how to use this template class to create linked lists to store student and course data.

2. **Student Class**:

   Create a `Student` class with the following attributes:

    - Student ID
    - Student name
    - GPA (Grade Point Average) Use the `LinkedList` template class to store a list of students.

3. **Course Class**:

   Create a `Course` class with the following attributes:

   - Course code
   - Course name
   - Instructor name Use the `LinkedList` template class to store a list of courses.

4. **Menu-Driven Interface**:

   Implement a user-friendly menu-driven interface that allows users to perform the following actions:

   - Add a student to the student list.
   - Add a course to the course list.
   - Display a list of students.
   - Display a list of courses.
   - Quit the program.

5. **Templates Usage**:
   - Utilize templates for the `LinkedList` class to ensure it can work with both the `Student` and `Course` classes.
   - Demonstrate the creation of instances of `LinkedList` for both student and course data types.

6. **Error Handling**:

   Implement error handling in your program using exception handling. For example, handle cases where the user tries to add a student to the student list but provides invalid input.

****