COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY
OBJECT ORIENTED PROGRAMMING – CT-260, SPRING 2023
ASSIGNMENT 2 – Group Assignment - CLO 2, MARKS 5
RELEASE DATE: 25-06-2023, DEADLINE: 09-07-2023

Guidelines:

- It is a group assignment. At maximum 2 students are allowed in a group. We have declared this assignment as a group assignment so that the members in a group can discuss about different ideas and implementation scenarios and come up with the best solution. We also aim to enhance the team management, planning, communication, and collaborative skills of students.
- 1 Mark on each question. In total there are 4 questions. 1 Mark is on writing proper test programs, proper submission of all the files, with required format by following the submission guidelines, before the deadline.
- In case of missing details or confusion, you can make assumptions. However, your assumptions must not contradict with the actual statements.
- Submission must include:
    - Code files (.cpp) zipped in a folder. Each file should have name with student ID (one of the group members), Assignment and Question number. For example: “CT-22034-A2-Q1.cpp”, “CT- 22034-A2-Q2.cpp” format.
    - A pdf file which includes output screenshots for each question. You can take screenshot by pressing windows + print screen buttons or you can use snipping tool. You can have multiple screenshots for a single question. You can store all screen shots in a word file and then can save it as a pdf. The pdf file must include the details about distribution of tasks among the group members and how would you rate the assignment content.
    - Each output must contain your ID and name on the screen.
- Plagiarism is punishable with zero marks in the task.
- Late submissions are not allowed.

---

1. You have been tasked with developing a library system for a university by incorporating all main pillars of object-oriented programming (OOP) while implementing logical functionality for managing books, library members, and borrowing transactions. Consider the following requirements for the university library system:
   a. Encapsulation:
      - Design classes to encapsulate the core entities of the library system, such as "Book," "LibraryMember," and "Transaction."
      - Implement private data members to store sensitive information like book details, member information, and transaction history.
      - Define appropriate public member functions to interact with and manipulate the library data securely.
   b. Inheritance:
      - Utilize inheritance to create specialized book classes, such as "Textbook" and "Novel," derived from the base class "Book."
      - Implement additional derived classes as needed, considering unique attributes and behaviors associated with different book types.
      - Utilize inheritance to create a class hierarchy for library members, such as "StudentMember" and "FacultyMember," derived from the base class "LibraryMember."
   c. Polymorphism:
      - Implement virtual functions in appropriate classes to enable polymorphic behavior.
      - Demonstrate runtime polymorphism by overriding virtual functions in derived classes, allowing for different behavior based on the actual object type.
      - Utilize function overriding and dynamic binding to provide specialized functionality for specific book types or member categories.
   d. Abstraction:
      - Design abstract classes, interfaces, or pure virtual functions to represent common behaviors or attributes shared by multiple classes.
      - Implement abstraction to define generic methods or interfaces that can be used by different book or member classes.
      - Utilize abstraction to hide implementation details and expose only essential information and functionality to the external components of the program.

   In addition to incorporating the main pillars of OOP, the program should include logical functionalities and interactions, such as:
      - Adding, removing, and updating books in the library inventory.
      - Registering new library members and managing their borrowing privileges.
      - Allowing members to borrow and return books, considering availability and loan limits.
      - Tracking borrowing history and generating reports on overdue books and popular titles.
      - Applying appropriate business rules, such as calculating late fees and enforcing return deadlines.
2. You have been assigned to develop a scientific calculator that supports various mathematical operations. The calculator should utilize class templates and function templates to handle different data types, while also providing specialized versions for specific mathematical operations. Consider the following requirements for the scientific calculator:

a. Class Template: Calculator
    - Design a class template called "Calculator" that represents a scientific calculator.
    - Implement member functions for overloading basic arithmetic operators (+, _ , *, /) on different data types.
b. Function Template: Power
    - Implement a function template called "Power" that calculates the power of a given number.
    - The function should work with numeric data types.
    - Ensure the function handles both positive and negative powers.
c. Specialized Class Template for processing matrix: Calculator
    - Create a specialized class template that represents a calculator specifically designed for matrix operations.
    - Implement member functions for overloading basic binary operators to perform matrix addition, matrix subtraction, and matrix multiplication.
    - Implement a member function for obtaining determinant of a matrix.
    - Ensure the class template works with matrix data types by using template specialization.
d. Specialized Function Template for processing float data types: Power
    - Develop a specialized function template for "Power" that calculates the power when the data type of base and power is float.

3. You have been tasked with developing an online voting system for a university's student council elections. The system should utilize object-oriented programming principles and incorporate exception handling to ensure the integrity and reliability of the voting process. Consider the following requirements for the online voting system:
    a. Class: Candidate
        - Design a class called "Candidate" that represents a candidate running for an election.
        - Implement member variables to store the candidate's name, party affiliation, and vote count.
        - Define member functions to update the candidate's vote count and retrieve the candidate's details.
    b. Class: Election
        - Create a class called "Election" that manages the overall election process.
        - Implement member variables to store the list of candidates and the total number of votes cast.
        - Define member functions to add candidates, cast votes, and retrieve election results.
    c. Exception Handling: InvalidVoteException
        - Define a custom exception class called "InvalidVoteException" that will be thrown if an invalid vote is cast.
        - Implement appropriate member functions to retrieve and display the exception message.
    d. Exception Handling: ElectionException
        - Define a custom exception class called "ElectionException" that will be thrown for general election-related errors.

- Implement appropriate member functions to retrieve and display the exception message.

Provide your solution by writing the necessary code for the classes mentioned, including any necessary member variables, methods, and exception handling mechanisms. **Note:** You can assume that the specific attributes and methods of the classes are left to your discretion.

4. You have been assigned to develop a data container for a student database system. The container should incorporate all the concepts of object-oriented programming, including class design, inheritance, polymorphism, templates, and exception handling, to provide a robust and flexible solution for managing student records. Consider the following requirements for the student database container:
   a. Class Hierarchy: Person and Derived Classes
      - Design a base class called "Person" that represents a generic person.
      - Implement derived classes for different types of people, such as "Student," "Professor," and "Staff."
      - Each derived class should have specific attributes and methods related to the role of the person.
   b. Class Template: Database
      - Create a class template called "Database" that represents the container for storing student records.
      - Implement methods to add, remove, and search for student records.
      - Ensure the class template supports different data types by using template parameters.
   c. Exception Handling: InvalidDataException
      - Define a custom exception class called "InvalidDataException" that will be thrown if invalid data is encountered during operations on the database.
      - Implement appropriate member functions to retrieve and display the exception message.
   d. Polymorphism and Dynamic Memory Allocation
      - Utilize polymorphism by designing a common interface, such as a virtual function, for performing common operations on different types of people.
      - Use dynamic memory allocation (e.g., new and delete) to manage memory for storing student records.