Object Oriented Programming – Fall 22
(BS-CS-F22)
Lab-12 (Function Templates and Exceptions)
Lab Instructor: Mam Sanam Ahmad

Instructions:

- **Indent your code properly.**
- **Use meaningful variable and function names.**
- **Use the camelCase notation.**
- **Use meaningful prompt lines/labels for all input/output.**
- **Do NOT use any GLOBAL variable(s). However, global named constants may be used.**
- **This is an individual lab, you are strictly NOT allowed to discuss your solution with fellow colleagues, even not allowed to ask how is he/she is doing, it may result in negative marking. You can ONLY discuss with your TAs or with me. • Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.**

Do Validations on inputs where required otherwise 1 mark will be deducted for every wrong validation.

Total Marks 50

Task 01 [10 Marks]

Write a program that:

- Declare the function template for finding the sum of two arrays
- This function template takes three arguments two arrays of the same data type and the size of the arrays. It returns a new array that contains the sum of corresponding elements of the two input arrays
- In the main function, declare two arrays of different data types int and float
- Call the function template twice once with the int arrays and once with the float arrays
- Display the output array on the Console

Task 02 [15 Marks]

Write a program that:

- Declare the function templates to add, multiply and subtract two matrices of different data types.
- These function templates take two matrices of types T1 and T2, their size, and creates a new matrix of type T1 to hold the result
- Overload the function templates to add, multiply and subtract two matrices of the same data type
- These overloaded function templates take two matrices of type T, their size, and creates a new matrix of type T to hold the result.
- Show the result on the console

Task 03 [25 Marks]

Design and implement a C++ program to manage an inventory system for a small store using arrays. The program should handle various inventory operations such as adding, removing, and updating items, and must handle **error conditions gracefully using custom exception classes.**

1. Item Class:

Data Members:

- string itemName
- int quantity
- double price

Methods:

- **Constructor** to initialize item name, quantity, and price.
- Getters and setters for each attribute.
- **void updateQuantity(int newQuantity):** Updates the quantity of the item. Throws an exception if the new quantity is negative.
- **void updatePrice(double newPrice):** Updates the price of the item. Throws an exception if the new price is negative.

2. Inventory Class:

Data Members:

- Item items [MAX_ITEMS] where MAX_ITEMS is a predefined constant representing the maximum number of items the inventory can hold.
- **int itemCount:** Keeps track of the current number of items in the inventory.

Methods:

- **void addItem(const Item& newItem):** Adds a new item to the inventory. Throws an exception if the inventory is full or if an item with the same name already exists.
- **void removeItem(const string& itemName):** Removes an item from the inventory by name. Throws an exception if the item is not found.
- **void updateItemQuantity(const string& itemName, int newQuantity**): Updates the quantity of a specific item. Throws an exception if the item is not found or if the new quantity is negative.
- **void updateItemPrice(const string& itemName, double newPrice):** Updates the price of a specific item. Throws an exception if the item is not found or if the new price is negative.
- **Item getItem(const string& itemName) const:** Returns a copy of the specified item. Throws an exception if the item is not found.
- **void printInventory() const:** Prints all items in the inventory along with their details.

3. Custom Exceptions:

Define and use the following custom exception classes, derived from runtime_error:

- **InventoryFullException:** Thrown when attempting to add an item to a full inventory.
- **ItemNotFoundException:** Thrown when attempting to access or modify an item that does not exist.
- **InvalidItemOperationException:** Thrown when an invalid operation is attempted on an item, such as setting a negative price or quantity.

4. Main Function:

Demonstrate the functionality of the inventory system by:

- Adding a few items to the inventory.
- Performing updates on item quantities and prices.
- Removing items from the inventory.
- Handling exceptions and displaying appropriate error messages.
- Printing the inventory list before and after performing operations.