**Lab 5**

***Topics covered: Templates and Exceptions***

**Objective:** The objective of this Lab is to examine C++ templates and exception handling.

**Task 1:** Create a class that will be used as the basis for the remainder of the lab.

1. Create a new project. You can **name this whatever you like**.
2. Design a class to abstractly model a **shelf of games**.
   1. **Define an array** (use the board game class defined in Lab 4 as the data type for the array). The total number of board games is kept in a global constant. You may choose a number, say, 10.
   2. Define a variable to hold the **number of board games currently kept in the shelf**.
   3. Create a **default constructor** that initializes any values so the shelf starts empty.
   4. Provide functions to **add** and to **remove** a board game. The add function should receive a board game as an input parameter and the remove should return a board game. For the sake of simplicity, assume board games are fungible.
   5. Provide a **function that returns the number of board games** currently being housed in the shelf.
   6. Create the implementation code for the above functions as required.
3. Include in the submission a description of how you designed, your, add and remove methods. This description should not be code or pseudo code. Complete this before moving on to task 2.

**Task 2:** Use the shelf class.

1. Create a program that tests the shelf class. Prompt the user for which class method to invoke. This may look like:

|  |
| --- |
| Press 1 to add a board game to the shelf.  Press 2 remove a board game from the shelf.  Press 3 see how many board games are currently on the shelf.  Press 4 to quit. |

* 1. When adding a board game to the shelf, dynamically create (using the new operator) the board game. You may want to set attribute values in the board game class.
  2. When removing a board game, you may want to display the attribute values set when the board game was created.

1. Use your test program to test all member functions and ensure the class is working correctly. At this time, **only test normal operating conditions that should not throw an error**, like no situations such as adding a board game when the shelf is full, or removing a game when it is empty.
2. Include in the lab report a screen shot(s) board of the output of your tests.

--- Milestone ---

**Task 3:** Throw exceptions and catch errors.

1. Modify the code in your shelf class to test if the shelf is full or empty.
2. Modify your main program so that it traps any errors and displays an error message to the user then continues operations normally. You will need to create the **exception classes**.
   1. If you are adding a board game to a full shelf, throw a **FullShelf** exception.
   2. If you are removing a board game from an empty shelf, throw an **EmptyShelf** exception.
3. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of the advantages of trapping an error in the class versus the calling function.

**Task 4:** Templates.

1. Create a new project
2. *Copy the code* from your shelf class into a new class called **EntertainmentCollection**.
3. Modify your class to use a **template** rather than board game as the item being stored in the array.
4. Copy and modify the code from task 2 to implement the EntertainmentCollection using the video game class created in lab 4.
5. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of the advantages of using a template over specific types.

**Lab Submission:**

1. Write a lab report including the following information:
   1. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.
   2. The sections from each task indicated to be included in the lab report.
2. Include all source code from all tasks, input and output files (if any), and any special instructions to compile and run those programs.

**Lab Grading:**

1. 20% - Lab attendance
2. 20% - Task 1 has been correctly implemented and meets all requirements.
3. 10% - Task 2 has been correctly implemented and meets all requirements.
4. 15% - Task 3 has been correctly implemented and meets all requirements.
5. 15% - Task 4 has been correctly implemented and meets all requirements.
6. 20% - Lab report contains all required information and is well written.

If program fails to compile, 0% will be given for that Task.