Lab 02 - Arrays

Instructions:

- **A point-of-sale (POS) system manages transactions, inventory, and customer data. Its inventory management application updates stock levels after sales or restocking, detects low-inventory items, and computes total inventory.**

- **Your objective is to construct an inventory management system that can:**
  - **adjust the count of an item.**
  - **generate reports of inventory items below a threshold.**
  - **calculate inventory count.**

  You must store product data (count) in *Array* object from Note03, and let the product ID be its index. The application must adhere to the following rules:
  - **Only a single item can be adjusted at a time.**
  - **The product ID identifies the item.**
  - **An item's count can never be negative.**

- **Your source codes must compile and can only include the libraries '`iostream`', '`iomanip`', '`string`', '`sstream`', '`fstream`', '`stdexcept`', '`Array.h`' and '`Object.h`' from Note03, and user-defined libraries from the lab to receive any credit**

- **A cumulative task will not receive credit if the required previous tasks are not completed.**

- **Your submissions must be submitted to the GitHub repository in the Lab02 directory.**

- **Cheating of any kind is prohibited and will not be tolerated.**

- **Violating or failing to follow any of the rules above will result in an automatic zero (0) for the lab.**

Grading

Task	Maximum Points	Points Earned
0	0.5	
1	2.4	
2	0.6	
3	1.5	
Total	5.0	

Note: solutions will be provided for tasks colored blue only.

Task 0 requires completing the Array class for Note03

Task 1

- Copy '`Array.h`' and '`Object.h`' from your Note03 directory to your Lab02 directory and create a header file name '`Inventory.h`' that create a class named *Inventory* that publicly inherits *Object* in the namespace *dsl* that contains:
  - a private integer *Array* field named *items*.
  - its public special member functions such that *items* has a default size of 100 with each element initially zero.
  - a public integer constant method that takes a product ID as a parameter and returns the count of the item associated with the product ID if the product ID is valid; otherwise, it throws an invalid-argument exception.
  - a public void method that takes a product ID and an integer as parameters and assigns the item associated with the product ID the value of the integer parameter only if the product ID is valid and the integer parameter is not negative.
  - a public string constant method that takes an integer as a parameter and returns a string that is a list of product ID (as two-digit values) of items that have counts less than or equal to the parameter if the parameter is not negative; otherwise, it returns an empty string.
  - a public integer constant method that takes an integer as a parameter and returns the number of items whose counts are less than or equal to the parameter if the parameter is not negative; otherwise, it returns 0.
  - a public integer constant method that takes no parameters and returns the total number of items in the inventory.
  - a public overridden `toString()` method that displays a list of each item in the inventory in the format:

    ID`:` *x*

    where *ID* and *x* are the product ID as a two-digit value and the item's count, respectively.

Task 2

- Within the header file '`Inventory.h`' and the namespace *dsl*, define the Boolean function named `Import()` that takes an *Inventory* reference and a string as parameters. It opens a file whose name is the string parameter, stores the data from the file in the *Inventory* parameter, and returns true if the file opens; otherwise, it returns false.

Task 3

- Create a C++ file named '`main.cpp`' that defines a void function named `app()` that takes an *Inventory* reference parameter. It displays a menu that provides a quit option and an option to invoke each method (excluding special member functions) of the *Inventory* class. For each option, it receives the necessary information and displays an informative output. The function should loop until the quit option is chosen.

  Afterward, test `app()` with an *Inventory* object that is preloaded with the content from the provided file '`data.txt`' by invoking `Import()`.

Extra Credit

- Create a header file named '`Extra.h`' that defines a modified *Inventory* class that includes the cost of items and methods to view the cost of an item, adjust the cost of an item, and retrieve the cost of the entire inventory. (1 point)