COMP 233 – Data Structures and Algorithms
Assignment 2 – Class Definition and Implementation
Due by 9:10PM, October 17

Write a class definition and implementation for a class that will model an item of inventory carried by a store.

Each item will have the following three attributes:

1. a part number. All part numbers will consist of two uppercase letters (A-Z) followed by four digits (0-9). Example: BN3782
2. a department code. All department codes will consist of three uppercase letters. Example: ELE
3. a price. All prices will be decimal numbers that are less than or equal to 999.99 and greater than or equal to 0.01 All prices will have two decimal places. Example: 87.25

NOTE: The above information is to enable you to define the data members of the class Item appropriately. You do ***NOT*** have to do any validation to make sure that the data actually confirms to the above rules.

- The class name ***MUST*** be Item (with a capital I.)
- Your class ***MUST*** be well encapsulated (private data members, public member functions.)
- The class definition ***MUST*** be located in a header file called Item.h, while the implementation of the member functions ***MUST*** be located in a source code file called Item.cpp.
- All member functions (other than constructors that consist of only initialization expressions and an empty body) ***MUST*** be implemented in the .cpp file as opposed to inside the class definition in the header file. Any constructors, which consist of only initialization expressions and an empty body, may be defined inside the class definition in the header file.
- Your header file ***MUST*** include an inclusion guard in the header file to prevent the possibility of multiple inclusions in a compilation unit.
- Your header file ***MUST NOT*** import the entire std namespace with a using directive. As an example in the case of string, either use the full name such as std::string in your header file or use a using directive that only imports string from the std namespace.
- All member functions, which do not change the values of the data members, ***MUST*** be declared as constant member functions.
- You may choose any convenient name and type for the private data members, but your class ***MUST*** include the following public member functions:
  NOTE: in all of the following parameter lists, the variable name is included here only to indicate what information is intended to be sent in that parameter in a function call; you may use whatever name you like for the parameter variable names. All strings are standard library strings as opposed to C-style strings.

  1) a default constructor, which will result in the partNum being set to a string of six dashes, the department code being set to a string of three dashes, and the price being set to 0.
  2) a three parameter constructor where the three parameters will be string partNum, string dept, string priceStr. You may assume that using stod on the priceStr will not fail.
  3) a three parameter constructor where the three parameters will be string partNum, string dept, double price.
  4) a one parameter constructor where the parameter will be string line, where the format of line will be "BN3782 ELE 87.25" for the above item. You should assume that there might be leading or trailing whitespace as well as possibly various amounts of white space between the three items, but you can assume that there will not be any whitespace in the part

number, the department code, or the price themselves. You may assume that using stod on the portion of the string with the price will not fail.

5) an accessor member function called getPartNumber( ) which will return the part number as a string
6) an accessor member function called getDepartmentCode( ) which will return the department code as a string
7) an accessor member function called getPrice( ) which will return the price as a double.
8) an accessor member function called getData( ) which will return a string in the format "BN3782 ELE 87.25" where the three items will be separated by one space and the price will be shown with no less than and no more than 2 decimal places.
9) a mutator member function called setPartNumber(string partNum) whose return type will be void.
10) a mutator member function called setDepartmentCode(string dept) whose return type will be void.
11) a mutator member function called setPrice(string priceStr) whose return type will be void. You may assume that using stod on the priceStr will not fail.
12) a mutator member function called setPrice(double price) whose return type will be void.
13) a mutator member function called setData(string line) where line will have the same format and characteristics as in the above one-parameter constructor and where the return type will be void. You may assume that using stod on the portion of the string with the price will not fail.

- Additionally, your class definition and implementation file ***MUST*** include the declaration and implementation of the following two operator overloads:

  NOTE: These will NOT be member functions; they ***MUST*** be friend global (i.e., non-member) functions. Both of these ***MUST*** work with any stream of the appropriate type (console, file, string streams, etc.) and should support chaining of the stream operators, such as cout << i << endl; and cin >> i1 >> i2;

14) An overload of the stream extraction operator >> where the function will extract information about the item from the stream in the form of "BN3782 ELE 87.25" where again there may variable amounts of whitespace before, between, and after the three values.
istream & operator>>(istream & in, Item & i)
15) An overload of the stream insertion operator << where the function will insert information into the stream in the format "BN3782 ELE 87.25" where the three items will be separated by one space and the price will be shown with no less than and no more than 2 decimal places.
ostream & operator << (ostream & out, const Item & i)

Finally, you ***MUST*** plan and implement test code in the main function of another .cpp file so that you can test and verify that ALL of your member functions work appropriately.

Submit the following prior to 9:10PM on Wednesday, October 17:

1) paper printouts of
   a. Item.h
   b. Item.cpp
   c. Your testing source code file
2) a compressed folder file containing all of the above items via Blackboard.