DSA/CS 5005 – Computing Structures – Programming Project – 1 – Summer 2016 (Due: June 27, 2016 11:59 PM)

Objectives

1. Use basic input/output (cin and cout) [10%]
2. Use arrays in your program [10%]
3. Create at least 2 working C++ classes [10%]
4. Use "exception handling" in your program [10%]
5. Design and implement the program according to the project description. 50% distributed as follows:
    a. Working class methods [20%]
    b. Correctness of the output from the given main program [30%]
6. Use STL vector class in your program [10%]
7. Document your project thoroughly as the examples in the textbook. This includes but not limited to header comments for all classes/methods, explanatory comments for each section of code, meaningful variable and method names, and consistent indentation. Program that lack or weak in documentation will receive a *deduction* of up to 30% of the grade.

Project Description

Google Taxonomy is used to categorize product items based on attributes/properties of the same. In this project, you will develop a C++ program to process products that adhere to google taxonomy requirements. The main purpose of this project is to get familiar with C++ and its features.

An input file will contain product details. There is unknown number of product items in the file. The details of consecutive products are separated by two stars "**". You have to read in each of the product details, and create objects to store them. All the product objects are stored using an array.

You will implement the following two classes.

The *base class* is called **Product.** The Product class has the following data members:
1. ID: An identifier for the product; consists of alphanumeric characters. For example: "mnst123us".
2. Category: This attribute indicates the category of the product according to the Google product taxonomy. For example, "Apparel & Accessories > Clothing Accessories > Sunglasses".
3. Condition: Condition or state of product i.e., either of "new", "used" or "refurbished".

The *derived class* is called **Item.** The Item class has the following data members:
1. Title: Tile of the product item. Example: "Kids Polo Shirt"
2. Availability: Availability status of the item, which can be any one of the following: "in stock", "out of stock" or "pre-order".
3. Color: The color of the item. For example: "red", "blue".
4. Price: The price of the item in USD. For example: "29.99" or "9.99".

You first read all the products into an array of Item objects. You should assume that you have 1000 Item information in the input file. The array should be declared as follows:

```
Item* MyItems = new Item* [1000]; //MyItems in pointing to an array of Item pointers
```

Every time you create a new Item object you will use the following statement:

```
Item* anItem = new Item();
```

Now you can store this newly created Item Object in MyItems as follows:

```
MyItem[0] = anItem;
```

After you store all the items in the array, you are required to find the output for the following queries.

1. The total number of products
2. Number of available products in each product condition types (new, used, and refurbished)
3. The title and price of the most expensive product item for each of the product conditions (new, used, and refurbished)
4. The title and price of the least expensive product item (new, used, and refurbished)
5. The average price of all the products

In this program you are required to measure the time taken in (millisecond) to complete the execution of the program. You can read the program execution time measurement shown in your text book (Section 2.4, Page 76) to learn more about *QueryPerformanceFrequency* and *QueryPerformanceCounter* for time measurement. The statements for execution time calculation should the first statement in the program and last statement of the program.

Now instead of using the array, you should modify your program (very slightly) so that we can use STL vector instead of the array. Your declaration will be something like this:

```
vector<Item*>* MyItems = new vector<Item*>();
```

Every time you create a new Item object you will use the following statement:

```
Item* anItem = new Item();
```

Now you can store this newly created Item Object in MyItems as follows:

```
MyItem.add(anItem);
```

Input File Format

A sample input file containing product details is given below:

```
mnst123us
Apparel & Accessories > Clothing
New
Mens Polo Shirt
In Stock
Black
39.99
**
dvd254ww
Media > DVDs & Movies
New
The Hunger Games: Catching Fire
Preorder
White
49.99
**
wmn555uk
Apparel & Accessories > Shoes
New
Women's Boots
In Stock
Red
89.99
```

File Input

The input filename will be part of command line parameters. Here is an example program that will read lines of text from a file.

```
// reading a text file
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main ( int argc, char *argv[] )
{
  string line;
  if ( argc != 2 ) // argc should be 2 for correct execution
    // We print argv[0] assuming it is the program name
    cout<<"usage: "<< argv[0] <<" <filename>\n";
  else {
    // We assume argv[1] is a filename to open
    ifstream myFile ( argv[1] );
    // Always check to see if file opening succeeded
    if ( !myFile.is_open() )
      cout<<"Could not open file\n";
    else {
                    while ( getline (myFile,line) )
                    {
                           cout << line << '\n';
```

```
                                  }
                                  myFile.close();
                     }
            }
    }
```

Constraints

1. This project must be implemented in C++. The headers you can use are <iostream>, <vector>, <fstream> and <string>. We will not use any other libraries.
2. None of the projects in DSA 5005 will be a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.
3. The project is due on June 27, 2016 by 11:59pm. Source file(s) should be submitted to the digital drop box. Multiple source files must be submitted in a single zip file.

Late Penalty

Submit your project on or before the due date to avoid any late penalty. A late penalty of 10% per day will be imposed after the due date. After five days from the due date, you will not be allowed to submit the project. If your submitted project doesn't work (or compile), you will lose 50% of the project grade.