**CpSc 1070   
Program 4**

Due Date: Thursday, April 26.

This is the final programming assignment of the semester! You may work with a partner. If you choose to work with a partner, **one** of you should send me an email by Friday, April 20, letting me know the name of your partner. All files that you write must contain the name and the username of both you and your partner; however, only **one** of you will submit the assignment.

**Problem Statement:** For this program assignment, you will be developing a program in support of a Dessert Shoppe which sells candy by the pound, cookies by the dozen, ice cream, and sundaes (ice cream with a topping). Your program will be used for the checkout system.

To do this, you will implement an inheritance hierarchy of classes derived from a **DessertItem** *abstract* base class.

The **Candy, Cookie, and IceCream** classes will be derived from the DessertItem class.

All monetary values must be of type ***Money*** and all weights must be of type ***Weight***.

The **Sundae** class will be derived from the **IceCream** class.

You will also write a **Checkout** class which maintains a list (Vector) of **DessertItems**.

**The DessertItem Class**

The **DessertItem** class is an *abstract* base class from which specific types of **DessertItems** can be derived. It contains only one data member, a name. It also defines a number of methods. All of the **DessertItem** class methods, except the *getCost()* method, are defined in a generic way in the file, [dessertItem.cpp](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog4/Public/dessertItem.cpp), provided for you, along with the other assignment specific files in the directory. The *getCost()* method is an *abstract*method that is not defined in the **DessertItem** class because the method of determining the costs varies based on the type of item. Tax amounts should be rounded to the nearest cent. For example, the calculating the tax on a food item with a cost of 1.99 cents with a tax rate of 2.0% should be 0.04. **DO NOT CHANGE THE DessertItem.cpp file!** Your code must work with this class as it is provided.

The [dessert.h](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog4/Public/dessert.h) file contains constants such as the tax rate, the name of the store, and the width used to display the costs of the items on the receipt. Your code should use these constant names wherever necessary!

**The Derived Classes:**

All of the classes which are derived from the DessertItem class must define a constructor. Please see the provided *main()* method in *driver.cpp*, to determine the parameters for the various constructors. Each derived class should be implemented by creating a file with the correct name, eg., *Candy.cpp*.

The ***Candy*** class should be derived from the ***DessertItem*** class. A ***Candy*** item has a weight and a price per pound, which are used to determine its cost. For example, 2.30 lbs.of fudge @ .89 /lb. = 2.05. The cost should be rounded to the nearest cent.

The ***Cookie***classshould be derived from the ***DessertItem*** class. A ***Cookie*** item has a number and a price per dozen, which are used to determine its cost. For example, 4 cookies @ 3.99 /dz. = 1.33. The cost should be rounded to the nearest cent.

The ***IceCream*** class should be derived from the ***DessertItem*** class. An ***IceCream*** item simply has a cost.

The ***Sundae*** class should be derived from the ***IceCream*** class. The cost of a ***Sundae*** is the cost of the ***IceCream*** plus the cost of the topping.

**The Checkout Class**

The *Checkout class* provides methods to do the following:

1. enter dessert items into the cash register,
2. clear the cash register,
3. get the number of items,
4. get the total cost of the items (before tax),
5. get the total tax for the items, and
6. print a receipt for the dessert items, including the total cost (cost + tax).

The Checkout class must use a Vector to store the DessertItems. The total tax should be rounded to the nearest cent. The complete specifications for the Checkout class are provided for you in checkout.h

**Compiling and Testing Your Program**

A simple testdriver, driver.cpp, along with its expected output, will be provided for you to test your class implementations. You can add additional tests to the driver to more thoroughly test your code.

1. Use the following command to build the executable:

*make*

1. Test the program using

./p4 < input.txt

**Submitting your files**

Create a tar file containing the five source files *Candy.cpp, Cookie.cpp, IceCream.cpp, Sundae.cpp, and Checkout.cpp*. *Candy.cpp,candy.h, Cookie.cpp, cookie.h, IceCream.cpp, iceCream.h, Checkout.cpp Sundae.cpp, and Checkout.cpp*.

**Requirements**

1. Your code must adhere to the [Program Standards](https://people.cs.clemson.edu/~rlowe/cs1070/programs/standards.html) and the [<="" a="">. Violations will lead to deductions.](https://people.cs.clemson.edu/~rlowe/cs1070/programs/requirements.html%3EProgram%20Requirements)
2. [<="" a="">Every method and class must be documented as described in the](https://people.cs.clemson.edu/~rlowe/cs1070/programs/requirements.html%3EProgram%20Requirements)[Program Standards](https://people.cs.clemson.edu/~rlowe/cs1070/programs/standards.html).
3. The documentation for each class must contain the the basic description of the class.
4. The documentation for each method must contain a basic description for the method, a very brief of each parameter (if names are descriptve, you can omit the parameter descriptions), and a brief description of the for any non-void returns, similar to the following example:
5. /\*\*
6. \* Inserts the integers in the other IntegerList in this Integer List
7. \* beginning at the given index
8. \* param other // other IntegerList
9. \* param n // index in IntegerList where first integer in the
10. \* // other IntegerList should be inserted
11. \*/
12. void insert(IntegerList other, int n);

**Getting the files for program 4**

1. Use *mkdir* to create a directory for program 4
2. **cd** to the prog4 directory.
3. Download the [prog4.tar](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog4/Public/prog4.tar) file to your prog4 directory.
4. Use the command below to extract the files in prog4.tar

***tar -xvf prog4.tar***

After you untar the files, you may delete prog4.tar

As always, you want to complete the assignment in a modular fashion. A makefile is provided for your convenience, but you will have to write skeletal files or comment out the dependencies that you are not ready to test, or delete what you are not ready to test, and add it back in as you complete various classes.

**Suggested Steps for Completing the Assignment**

1. Copy input.txt to a test file that will be a subset of input.txt.
2. The first number in the test file represents the number of customers. A semicolon is used to separate customers. Change the number of customers to 1, then
3. Delete dessert items, leaving only two Candy items.
4. Code and test the *Candy* class
5. partially code the Checkout class, enough to read two Candy items, then print the receipt for the two items.
6. Create the .h files for the classes that you are to implement.
7. Create skeletal files for Cookie.cpp, IceCream.cpp, and Sundae.cpp
8. Modify the Checkout class to handle reading and printing a Cookie item.
9. Code and test the *Cookie* class. It is very similar to the Candy class.
10. Add one cookie to the test file.
11. Code and test the *IceCream* class. It is simpler than the Candy
12. Finally, complete the code.

**Submitting your files**

1. If you worked with a partner, **only one** of you should submit the assignment. All files should contain the name and username for both of you.
2. First, if you are not in your program 4 directory, move to it and *make clean*. Make sure that you also delete *prog4.tar*.
3. tar all files to  myprog4.tar:    *tar  -cvf  myprog4.tar  \**   
   There is a 10-pt penalty for using the wrong filenames, and a 10-pt penalty for submitting to the wrong folder.
4. Submit ONLY myprog4.tar via [handin](https://handin.cs.clemson.edu/)

It is your responsibility to make sure you have submitted the correct files to the correct handin folder. Check your Handin folder after submitting the file. Again, there is a 10-point penalty for using the wrong file name and a 10-point penalty for submitting the file to the wrong folder.