ComS 327 Final Exam, November 24, 2020

This is an open notebook, piazza, zybook, canvas final exam. That is, you may use resources you find inside these tools, but not Internet links that go outside these tools. You may also use your previous assignments. Do not talk about this test with others (in or out of class) or use the Internet as described above. If you have questions if something can be used, it is better to ask first. If you think it is sketchy, it probably is. Suspected cases of academic dishonesty will be forwarded to the Dean of Students, and if confirmed, a grade of F will be assigned for the course. You have 24 hours to complete this test which will be due at 6:30 pm on Wednesday, November 25. Since this is a programming test, package up your source files, make files, and other documentation files, and place them all in a zip file and submit them via the Canvas assignment associated with this document. **Note, even if your program does not compile or run, we will grade what you turn in for partial credit.**

A candy store owner keeps his inventory of candy organized in a flat text file using his own home brew format. This format is given here with each item on its own line of text in the file. Note also that there can be multiple candies, each enclosed in a START, END pair.

```
START
Candy Name
Candy Type
Price Type
Price
Color
Amount
Calories
END

START
.
.
.
END
.
.
.
```

The “Candy Name” is the retail name of the candy.
The “Candy Type” is the type of candy and may be one of the following: “jelly” “caramel” “chocolate” “hard”
The “Price Type” indicates if the candy is sold by piece or pound. Valid entries are “piece” and “pound”
The “price” is the cost of the item for the given price type.
The Color is the color of the candy. Only “jelly” and “hard” candies have a color defined. Valid colors are the numbers 0 through 9 with 0 indicating no color.
“Amount” is the number of candies in stock or ounces of the candy, depending on the Price

Type. (Note that there are 16 ounces in a pound)
"Calories" is the number of calories in one unit (piece or pound) of the candy.

Write a complete program that may be a mix of C and C++ that outputs the following information for the store owner subject to the following requirements. However, you must use objects to encapsulate data and methods where appropriate. **Significant points will be lost for writing a c program, or program that doesn't use objects.**

1. The program outputs the following information:
    a. A list of all candies that includes the name, the amount on hand in the store, and the total price of the each item's current inventory. (how much would it cost to buy out the store of all of a item) in sorted order from highest to lowest of the total value of the item in in stock in the store. Output of each candy and its information should be one item per line.
    b. A summary line that prints the total value of candy in all inventory in the store.
    c. The total number of different types of candy of each of the four types, i.e., number of jelly types, number of caramel types, etc.
    d. The file is arbitrarily long and you do not know how many items are in the input file ahead of time, nor is there a max number of items.
    e. The program runs by typing in the command: "candy <filename>" where <filename> is the name of the text file to input. The results are displayed to standard output.
    f. Your program must use objects with inheritance and polymorphism appropriately. If there is a good reason to use an object, then you should do so. If polymorphism can be used in a useful way in this project, then you should do so. (Note that there are places that you should use objects, and there are places that polymorphic calls make sense in this problem.)
    g. Your C++ classes must contain valid copy constructors, destructors, and overloaded assignment operators.
    h. Documentation is important, 25% of your points will come from documenting your code. Please over document rather than under document your code. Do not skimp. In addition to documenting lines of code, include good documentation for each function and method you write as a block comment before the definition of the function or method. In addition to the points given for writing good documentation, what you give in documentation will also help assign partial credit when it is unclear what your intent was in the code.
    i. Include in your project, a makefile that by default will produce an executable file, but also contains the targets "clean" that removes all automatically generated files. Your makefile should not compile or do operations that are not necessary.