Homework Assignment 2 – Due by March 19th, 10 AM

In this homework, you are going to write a program to represent various types of packages offered by package-delivery services. You need to create an inheritance hierarchy to represent various types of packages with different shipping options and associated costs. The base class is Package and the derived classes are FlatRatePackage and OvernightPackage.

The base class Package includes the following attributes:

- Sender information: *senderName* (string), *senderAddress* (string), *senderCity* (string), *senderState* (string) and *senderZIPcode* (long).
- Recipient information: *recipName* (string), *recipAddress* (string), *recipCity* (string), *recipState* (string) and *recipZIPcode* (long).
- Other package details: *label* (string), *date (string)*, *weight* (double), *costPerOunce* (double), *insuranceType* (string), and *signatureConfirmation* (string). The attributes *label*, *date*, *weight*, *costPerOunce* represent the package label, the date the package is shipped, the package weight, and the standard shipping fee per ounce, respectively. The *insuranceType* attribute describes the extra package insurance option. A sender can purchase insurance coverage for mailpieces for up to $5,000 to protect against loss or damage. If the sender does not purchase insurance, then *insuranceType* is set to “none” and no extra charge is incurred. If the sender prefers to insure the package, then there are two options based on the package’s declared value or desired coverage: up to $1,000, or up to $5,000. *insuranceType* can thus have the values: “none”, “upto1000”, or “upto5000”. The attribute *signatureConfirmation* describes another extra service. A sender may request that the receiver signs upon receipt of a package, which incurs an additional charge. This attribute can thus have one of two values, either “none” or “sign”.

The specialized class FlatRatePackage has four attributes of its own:

- *type* (string), *width* (double), *length* (double), and *height* (double). The attribute *type* refers to the package type, which could be a “letter”, a “paddedEnvelope”, a “legalEnvelope”, or a “box”. The attributes *width*, *length*, and *height* refer to the package dimensions. The package dimensions are particularly important in deciding if a given mailpiece should be sent in a small box, a medium one, or a large one.

The specialized class OvernightPackage has two attributes of its own:

- *overnightFee* (double) and *tracking* (string). The attribute o*vernightFee* represents an additional fee per ounce charged for overnight-delivery service. The attribute *tracking* represents an additional service charged for providing detailed tracking information of the package. Its value is set to either “none” if the sender does not want to track the package or “track”, in which case an additional charge will be incurred.

Each class must have the following member functions:

- a constructor to initialize and validate all the corresponding attributes. For example, you need to ensure that weight, cost, fees all have positive values, and box dimensions are within the dimensions given in the table below.

- set and get functions to assign values and retrieve attribute values (**as needed**) as well as print function to print the package's attributes on the screen (specialized and inherited), and its shipping cost.
- its own calculateCost function that calculates and returns the cost associated with shipping a certain package.

  The base class Package's calculateCost function should determine the cost by multiplying the *weight* by the *costPerOunce* and adding the insurance and signature confirmation fees if applicable. Signature confirmation costs an additional $2.90. Insurance fees are based on the item's declared value or *insuranceType* as shown in the table below:

Insurance Type	Insurance Fee ($)
Upto1000	5.25
Upto5000	5.50

  FlatRatePackage should redefine calculateCost function so it calculates the shipping cost based on the package type and dimensions and independent of its weight. A FlatRatePackage's cost is a flat fee that depends on the package type and dimensions as summarized below. CalculateCost should also take into account extra services costs such as insurance and signature confirmation fees if applicable.

	Width	Length	Height	Cost ($)
Box	< 12	< 12	< 5	20.10
Box	< 11	< 8	< 5	17.75
Box	< 8	< 5	< 2	11.30
PaddedEnvelope				9.45
LegalEnvelope				7.65
Letter				3.20

  OvernightPackage should redefine calculateCost function so it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. CalculateCost should also take into account extra services costs such as insurance, signature confirmation, and tracking fees if applicable. Tracking incurs an additional charge of $5.50 per package.

After you design your base and derived classes, you need to design a PackageInventory class that keeps track of all packages sent.

The PackageInventory class has the following attributes:

```
vector <Package>  packages;
vector <FlatRatePackage> flatRatePackages;
vector <OvernightPackage> overnightPackages;
```

The PackageInventory class has a constructor as well as the following member functions:

- CreateNewPackage that creates a Package object, pushes it into the packages vector, and displays a message that the package was successfully added. If any package attribute value is not valid, you should not add an object to the vector and should print appropriate error message.

- CreateNewOvernight that creates an OvernightPackage object, pushes it into the OvernightPackages vector, and displays a message that the package was successfully added. If any package attribute value is not valid, you should not add an object to the vector and should print appropriate error message.

- CreateNewFlatRate that creates a FlatRatePackage object, pushes it into the FlatRatePackages vector, and displays a message that the package was successfully added. If any package attribute value is not valid, you should not add an object to the vector and should print appropriate error message.

- PrintAllPackages that prints the details of all packages in the packages vector: sender and receiver information, package label, date, weight, insurance option, signature confirmation requirements if any, and shipping cost.

  Note: when printing sender and recipient information (name, address, and city), you get extra credit if you print spaces between words. Each new word starts with an uppercase letter. For example:

```
Sender:     Lou Brown
            1 Main St
            Boston, MA 11111
Recipient:  Mary Smith
            7 Elm St
            New York, NY 22222
Label:      FY0002
Mailed on:  12/03/2016
Weight:     8.5
Insurance:  none
Signature:  none
Cost:       $4.25
```

- PrintAllFlatRatePackages that prints the details of all packages in the FlatRatePackages vector in a neat fashion: sender and receiver information, package label, date, weight, type, dimensions, insurance option, signature confirmation requirements if any, and shipping cost.

- PrintAllOvernightPackages that prints the details of all packages in the OvernightPackages vector in a neat fashion: sender and receiver information, package label, date, weight, insurance option, signature confirmation, tracking option, and shipping cost.

- PrintShippedonMonthYear that prints the details of all base and derived packages shipped on a specific month and year.

- CalculateTotalCost that calculates and prints the total cost of a specific type of packages. This includes the shipping cost of all packages in a specific vector.

- ProcessTransactionFile that opens the transaction file and processes its lines one by one. If the file could not be opened, you should print appropriate message.

Note: Make sure to break up your classes into .cpp and .h files and declare any member function that does not modify the attributes as constant.

After you design all your classes, you need to write a main program that displays the address information and calculates the shipping costs for several packages. The main program instantiates an object of type *PackageInventory* and calls a method to read a transaction file and process its commands. Your main should look like this:

```
int main()
{
      PackageInventory inventory;
      inventory.ProcessTransactionFile("TransactionFile.txt");
      return 0;
}
```

The transaction file contains several commands and corresponding values that will create new packages and add them to the corresponding vectors, invoke get functions to obtain the address information of the sender and the recipient, then print the two addresses as they would appear on mailing labels. Also, call each Package's calculateCost member function and print the result. The commands and their formats are:

- CreateNewPackage *senderName senderAddress senderCity senderState senderZIPcode recipName recipAddress recipCity recipState recipZIPcode label date weight costPerOunce insuranceType signatureConfirmation*

- CreateNewFlatRate *senderName senderAddress senderCity senderState senderZIPcode recipName recipAddress recipCity recipState recipZIPcode label date weight costPerOunce insuranceType signatureConfirmation type length width height*

- CreateNewOvernight *senderName senderAddress senderCity senderState senderZIPcode recipName recipAddress recipCity recipState recipZIPcode label date weight costPerOunce insuranceType signatureConfirmation tracking OvernightFee*

- PrintAllPackages

- PrintAllFlatRatePackages

- PrintAllOvernightPackages

- PrintShippedonMonthYear *month year*

- CalculateTotalCost *specificVector*

A sample transaction file is shown below. You may use it to test your code:

```
CreateNewPackage LouBrown 1MainSt Boston MA 11111 MarySmith 7ElmSt NewYork
NY 22222 FY0002 12/03/2016 8.5 0.5 none none

CreateNewPackage AmyJohnson 3465RegentsRd SanDiego CA 92130 EdwardJohnes
439NWGreens Fayetteville NY 13066 HG9983 06/03/2016 10 0.7 upto1000 sign

CreateNewPackage  BrianFlemings  4356NobelDr  SanDiego  CA  92130  MarySmith
1245ViaLaGitano Poway CA 93064 LM9983 10/09/2016 3 0.3 upto5000 sign

PrintAllPackages

CreateNewFlatRate  JuliaWu  4387StMary  Lafayette  LA  70504  DanielTessier
234JohnsonSt Lafayette LA 70503 JJ3981 12/30/2015 8.8 0.2 upto1000 sign
paddedEnvelope 10 5 1

CreateNewFlatRate  JuliaSmith  4587StMary  Lafayette  LA  70504  JimEdwards
256JohnsonSt  Lafayette  LA  70503  MJ3981  12/30/2015  5.8  0.2  upto1000  sign
LegalEnvelope 10 5 1

CreateNewFlatRate LisaKlein 5Broadway Somerville MA 33333 BobGeorge 21PineRd
Cambridge MA 44444 JJ8765 07/20/2014 10.5 0.65 upto1000 sign letter 2 2 0

CreateNewFlatRate  JaredBeth  6GenesseeRd  Hillsboro  OR  97124  AndreaSmith
65SquareBlvd Philadelphia PA 19121 PO0900 01/12/2015 12 1.1 upto1000 none box
5 6 3
```

```
CreateNewFlatRate  CarolynDavis  101DoveSt  SanDiego  CA  92129  BrianJames
8765HilcrestRd LasVegas NV 89116 ZT6751 09/14/2016 5 0.3 none none box 7 15
3

CreateNewFlatRate  MatthewGarcia  10876PowayRd  Poway  CA  93064  JennyFox
125BridleRd Manlius NY 13057 BV7800 12/01/2016 17 0.9 upto5000   sign box10
10 4

CreateNewFlatRate AmeliaPerez 908BrooksAve Houston TX 77001 MichaelBrown
3987BrieSt Beaverton OR 97076 RS1010 12/21/2015 5 0.2 none none box 3 10 5
```

```
PrintAllFlatRatePackages

CreateNewOvernight EdLewis 2OakSt Boston MA 55555 DonKelly 9MainSt Denver CO
66666 LL1322 03/08/2017 12.25 0.7 none none none 2.0

CreateNewOvernight  MaryPalmer  6534SpringburstDr  PalmSprings  CA  92240
DennisGarcia  8FifthSt  Denver  CO  66665  UI0900  10/11/2016  20.1  0.8  upto5000
sign track 7

CreateNewOvernight JoeGarcia 4320CaminoDr LaJolla CA 92037 JamieCliff
7655FrieserDr Bellevue WA 98005 KI0067 02/05/2017 9.3 3 upto1000 none track
5

CreateNewOvernight  MandyDidier  8764CreekViewBlvd  Fayetteville  NY  13066
SuzyWalter 8765RegentsRd SanDiego CA 92131 WV4544 12/01/2016 8.2 1.1 none
none track 8

PrintAllOvernightPackages

PrintShippedonMonthYear 12 2016

PrintShippedonMonthYear 2 2017

CalculateTotalCost packages

CalculateTotalCost flatRatePackages

CalculateTotalCost overnightPackages
```

Once done, you need to submit the following:

1. An electronic copy of your source code and executable file. You should place all source codes in a zipped folder and name it based on your first/last name and the assignment number. For example, JoeSmith-A1.zip. You should submit the zipped folder online through cougar website.