**MSCF Financial Computing II**

**Homework 5**

***Due At 12:30 pm Monday, Dec. 3, Before the Quiz***

***You will lose 10 points per hour after that time***

1. **(100 points) Python Basics**
2. **cme.20180320.c.pa2.zip** is a zipped CME SPAN file from March 20, 2018. In Programming Prep, many of you developed a program to process this file and report on WTI Crude Oil and Henry Hub Natural Gas futures and option contracts (Homework 1 part 1). This zipped SPAN file and my own C++ solution code to Homework 1 part 1 are included with this homework.

Your job is to create a Python program, in a file named **file\_processing.py**, that accomplishes the same task as the previous C++ program.

You Python program should read the March 20, 2018 SPAN file, and process contracts between July, 2018 and December, 2020, inclusive, rather than the date range we used in Programming Prep HW 1 part 1.

*You will have to learn how to format strings*, particularly **float** values, which we did not discuss in class. See https://docs.python.org/3.7/library/string.html

1. The file **EuroCallPrice.cpp** is my own C++ solution code to part of an FC I homework assignment in which you developed binomial tree code to price a European call option.

Your job is to create a Python program, in a file named **EuroCallPrice.py**, that accomplishes the same task as the C++ program. You will need to set up a nested loop to create a list of lists of **Price** objects, which you can use like the **vector<vector<Price>>** from the C++ program. (Next week, we will learn about *list comprehensions* that give a more compact way of accomplishing this.)

1. Define and test a **BinaryTree** class, in a file named **BinaryTree.py**, with the initial code below provided for you:

**class BinaryTree:**

**class \_BTNode:**

**def \_\_init\_\_(self, value, left = None, right = None):**

**self.\_value = value**

**self.\_left = left**

**self.\_right = right**

**def \_\_init\_\_(self):**

**self.\_top = None**

**def insert(self, value):**

**if self.\_top == None:**

**self.\_top = self.\_BTNode(value)**

**else:**

*… yours to finish …*

At the bottom of **BinaryTree.py** is testing code. You will need to finish the **insert**

method, to insert the value into the correct place in the tree (assume lower values are to

the left, and duplicate values are ignored/not allowed). You will also need to define

these **BinaryTree** methods:

**\_\_str\_\_** # return a string (**str**) representation of in-order elements, for

# example: '2 3 7 9 13'

**size** # return the number of elements in the tree

**depth** # return the (maximum) depth of the tree

**print\_pretty** # display the tree in “pretty” indented format, for example

# 13

# 9

# 7

# 3

# 2

***REMEMBER*** to put both authors’ names into your source code files.Put your **file\_processing.py**, **EuroCallPrice.py**, and **BinaryTree.py** files into a **.zip** archive and upload to the course web site.