CMPT 103 – Lab #8

General Information

Python version and IDE:	Python 3.3 / Wing IDE 101
Allocated lab time:	2 hrs and 50 min
Due date:	At the end of the lab period
Lab weight:	3%

Topics

- ✓ File I/O, tuples, lists, sets, and dictionaries

Submission

- ✓ **All code file (.py) should be submitted electronically** to your Lab Blackboard site.
- ✓ A portion of the total marks (20%) will be allocated for the programming style. For example, functions should be small; avoid writing duplicate code; names should be meaningful and descriptive; naming conventions should be followed consistently; code should be formatted properly; and comments should be accurate and well written.
- ✓ Comments are **required** for:
  - EACH program indicating the student name and program name.
  - EACH function indicating the function purpose, syntax (example usage of the function), parameters, and return value
  - Any block of code for which the purpose may be unclear (Note: you should always try to write clean code that can be understood easily without comments).

Assignment

For this lab, please put all functions into a file called `Lab8your_initials.py` (e.g., `Lab8FL.py` where F and L are the first letter of your first name and last name). Please feel free to write helper functions if necessary.

1. [40 marks] **Tuples and lists**

A stock portfolio can be viewed as a number of records, where each record has five fields related to a single stock: (a) the purchase date, (b) the purchase price, (c) the number of shares, (d) the ticker symbol, and (e) the current price. We can store these five fields in a tuple. Given a list of these tuples (a portfolio), we can do a number of simple operations.

Let's imagine that we have the following portfolio:

Purchase Date	Purchase Price	Shares	Symbol	Current Price
25 Jan 2001	43.50	25	CAT	92.45
25 Jan 2001	42.80	50	DD	51.19
25 Jan 2001	42.10	75	EK	34.87
25 Jan 2001	37.58	100	GM	37.58

We can represent this portfolio as a list of 5-tuples, where each 5-tuple contains the purchase date, purchase price, number of shares, ticker symbol, and current price.

```
portfolio = [('25-Jan-2001', 43.5, 25, 'CAT', 92.45),
             ('25-Jan-2001', 42.8, 50, 'DD', 51.19),
             ('25-Jan-2001', 42.1, 75, 'EK', 34.87),
             ('25-Jan-2001', 37.58, 100, 'GM', 37.58)]
```

Write a function named `total_purchase_price(…)`. This function must accept one argument (a portfolio in the format described above), examine each record in that portfolio, multiply the shares by their purchase price, and return the total purchase price of the portfolio.

Write a function named `total_value(…)` that takes a portfolio in the same format above, and returns the total current value of the portfolio.

Write a function named `total_gain(…)`.This function must accept one argument (a portfolio as described above), examine each record in that portfolio, compute the total amount gained or lost for that record, and return the total amount gained or lost for the portfolio. Use the `total_purchase_price` and `total_value` functions to implement `total_gain`.

Finally, write a test function named `testQ1` that demonstrates all of the three functions above. For example, given the above portfolio, the test function should display this output:

```
>>> testQ1()
    Total cost      = 10143.0
    Current value   = 11244.0
    Total gain/lost = 1101.0
```

2. [60 marks] **Sets, dictionaries, tuples, and lists**

   The provided text file, `yearly_temperature.txt`, contains all of the recorded temperatures for city in 2014.

   Implement the following functions that process the temperature data. Do not open, read, and close the file in each and every function; instead, **read the file once and build a suitable data structure with its contents** outside of these functions. **Provide this data structure to each of these four functions as an argument.** Your `demo` function (described later) must create this data structure for your lab instructor. **Include error handling when accessing a file.**

   a. `month_temp(…)`: this function accepts a month (string) and temperature data as arguments and returns a set of all the uniquely observed temperatures, i.e., the temperatures with no replication.

   b. `common_degrees(…)`: this function accepts temperature data and returns a set of the temperatures that occurred in all months of the year, i.e., the temperatures that are common to all months.

c. `rare_degrees(…)`: this function accepts a month (string) and temperature data as arguments and returns the set of temperatures that only occurred in this month and never occurred in any other month of the year.

d. `degree_limits(…)`: this function accepts temperature data and returns a dictionary: the keys of the returned dictionary are the months, and the value for each key (month) is a tuple containing the lowest and highest temperature of that month (in that order).

e. Finally, write a function named `testQ2` especially for your lab instructor. Calling this function should build the required data structure (from the file) and then demonstrate all of the four functions listed above.

```
>>> testQ2()
    Building the dictionary of temperature data: temps...

    month_temp("February", temps): {26, 27, 3, 22, 23}
    common_degrees(temps): {22, 23}
    rare_degrees("December", temps): {12}
    degree_limits(temps):
    April: (5, 40)
    January: (19, 23)
    February: (3, 27)
    June: (8, 36)
    August: (15, 49)
    July: (9, 49)
    November: (11, 49)
    May: (1, 36)
    March: (4, 33)
    December: (12, 49)
    September: (14, 49)
    October: (18, 49)
```