CS265 Introduction to Programming Laguages
Program 5

Objective
Work with files
Working with formatted output
Work with dictionaries/hashmaps

Assignment
These programs will store a collection of parts and their prices. Prices can be updated. Parts can be removed from the collection. You will need to write several functions/methods which report various information about the collection.

Write each of the following programs in C++, Python, and Java. Make sure to use the appropriate extension for each file (cpp, .py, and .java, respectively). Name each file prog5ABC.extension, where ABC are your initials. My 3 files would be prog5TLS.cpp, prog5TLS.py, and prog5TLS.java. Also include a comment section, with your name and contact information. **Programs with improper names and missing comment section will not be graded**.

Do not use global variables in any program.

Remember, unless the function specifically requests that you write information to the display, do not write anything to the display.

Make sure your function/method headers are written EXACTLY as specified below.

For the functions returning a Boolean value ensure that you return the appropriate value for the language: `true` for Java and C++, and `True` for Python.

Use the following declarations for the data structures:

```
unordered_map <string, double> parts //C++
HashMap <String, Double> parts       //Java
```

Required Functions

```
boolean addPart (HashMap <String, Double> parts, String part, double cost)
bool addPart (unordered_map <string, double>& parts, string part, double cost)
def addPart (parts, part, cost)
```
Ensures that the specified part/cost key/value pair is in the data structure. Returns a true value if the pair is a new pair/added to the data structure. Returns a false value if the pair is not new, with the cost being updated.

```
int totalParts (HashMap <String, Double> parts)
int totalParts (unordered_map <string, double> parts)
def totalParts (parts):
```
Returns the number of unique parts (int) in the data structure.

```
ArrayList<String> partsGreaterThan (HashMap <String, Double> parts, double upperLimit)
vector<string> partsGreaterThan (unordered_map <string, double> parts, double upperLimit)
def partsGreaterThan (parts, upperLimit):
```
Returns a list/array of the part names with a price greater than or equal to the `upperLimit` variable. The list should be created in the function.

```
boolean isPart (HashMap <String, Double> parts, String part)
bool isPart (unordered_map <string, double> parts, string part)
def isPart (parts, part)
```
Returns a true value if `part` is a valid part in data structure, otherwise return a false value. The case of the parts should not matter.

```
String leastExpensivePart (HashMap <String, Double> parts)
string leastExpensivePart (unordered_map <string, double> parts)
def leastExpensivePart (parts):
```
Returns the part name with the least expensive price. You can assume that there will be only one part with the least expensive price.

```
String mostExpensivePart (HashMap <String, Double> parts)
string mostExpensivePart (unordered_map <string, double> parts)
def mostExpensivePart (parts):
```
Returns the part name with the least expensive price. You can assume that there will be only one part with the least expensive price.

```
double averagePrice (HashMap <String, Double> parts)
double averagePrice (unordered_map <string, double> parts)
def averagePrice (parts)
```
Returns the average of all parts in the dictionary. Return -1 if you are unable to determine the average.

```
void printParts (HashMap <String, Double> parts)
void printParts (unordered_map <string, double> parts)
def printParts (parts)
```
This function WILL write to the display a table of each part and its price. Include column headers in the output. Make sure the price has two places after the decimal point. Also make sure the columns are neatly aligned, with the part column being left justified and the price column being right justified. This function/method should not return a value.

Questions?
Ask, sooner is better than later.