**COSC 1336, Lab 7 Instructions, Lists, Tuples**

**Part 1:** In a previous lab, you read data from a file, did some processing, and wrote data to a file. This lab will do similar work, but with more advanced processing. The data will be stored in a list where it can be easily processed with more advanced techniques. Once data is gathered in a list, a large variety of data processing tasks becomes much easier.

Write Python code that will: 1) read a text file into a list; 2) sort the list; and 3) write the sorted list to a different file. Here are some details:

Ask the user for a file name. If they did not include the .txt extension on the end, add it on yourself. (If they did include the .txt, do NOT append it. Avoid a double file extension: .txt.txt.) You can use **endswith** to see if the filename ends with “.txt”. You can use **+=** or **append** to add the “.txt” to the end of the filename if needed.

Open the file for reading, read the contents into a list. Sort the list, and write the sorted file out to storage. Put the word: “sorted” in the output name before the .txt extension. For example, if the user enters: people, notice that the filename does not end with .txt, so add the .txt and open the file people.txt. Sort the names (assume one name per line) then write to the filename:people\_sorted.txt.  
  
Provide a few status messages so the user knows the job was done. **Test your program on the data file: people.txt**, which is provided. Catch an exception if the file does not open properly and print out a useful error message. Example:  
  
 **Enter name of file to be sorted. “.txt” will be appended if missing: people  
 people.txt opened, xx lines found, sorted and saved to: people\_sorted.txt**  
  
For helpful code samples, look in the textbook: **“Working with Lists and Files”,** starting on page 373 (4ed), page 321 (3ed), page 325 (2ed). There are several programs that use files and lists. Programs 7-15 to 7-17 on pages 375 – 376 (4ed), pages 322 – 324 (3ed), or 8-15 to 8-17on pages 326 – 328 (2ed) do a lot of the work, but do not sort, catch exceptions, or modify the file names.   
  
If desired, temporarily save this part to the file: **DDHH\_L7\_Lastname\_Sort.py**

**Part 2:** The people.txt data would be more useful if you could search to see if a name is in the list. Also, it would be nice to add and delete names from the list. Using insert, append, or +, you can add new items to a list. Using del you can remove an item from a list. Using in and index, you can search to see if an item is in a list.  
  
Create a function called **insertName**. The IPO is:  
 Input: namelist, name. (namelist is the list of names, name is a short string)  
 Processing: If name is not already in the list, insert it.  
 Output: modified namelist

Create a function called **findName**. The IPO is:  
 Input: namelist, target. (namelist is the list of names, target is a short string to search for)  
 Processing: If target is in the list, return True, if not, return False  
 Output: boolean True or False depending whether target is in the namelist

Create a function called **deleteName**. The IPO is:  
 Input: namelist, target. (namelist is the list of names, target is a short string to delete)  
 Processing: If target is in the list, delete it from the list  
 Output: modified namelist

It is possible to write this without any functions, but I’d like you to get more practice writing and calling small function.

From part 1, your program can: read a file, sort a file, and write a file. With these new functions, you have a full-featured database application. To make this easy to use, provide a menu with a user’s choice driven loop. In the loop, ask the user what to do, and do it, until the user wants to quit. Here is pseudo code. It will be implemented in main().

Loop until quit:

**Enter choice: r)ead, w)rite, i)nsert, d)elete, f)ind, s)ort, v)iew, q)uit:**  
if choice is r for read:  
 ask for the filename, open the file, read the file into a list  
else if choice is w for write:  
 write list to the file and close the file  
else if choice is i for insert:  
 ask for a new name to insert, call insertName  
else if choice is f for find:  
 ask for a name to find for, call findName, output result (is in list, is NOT in list)  
else if choice is s for sort:  
 sort the list  
else if choice is v for view:  
 view the list (display the contents of the list namelist to the screen)  
 To save on output display screen space, display 10 names per line  
else if q for quit:  
 close the file, say goodbye, exit the loop, the program stops

In part 2, keep the input file name (people.txt, or whatever the user enters during the ‘r’ option) separate from the output file name (written during the ‘w’ option). To create the output file name, modify the input file name. If the input file name is: “people.txt”, modify the name to create the output file name: “people\_out.txt”. This avoids over-writing the input file name. It also pairs input files with output files. Provide a more flexible approach: allow the user the option to provide an output file name. If a name is entered, append .txt if not already there. If the name is NOT entered, use the modified input file name as a default. These “nice touches” make the program easier to use.

For example, if user selects option w for write, show this prompt:  
**What file to write output to (enter nothing for people\_out.txt; .txt added if needed)?**

You can move the operations to read, write and sort a file into functions. Test this on people.txt. Add a few names, delete a few names, find a few names, sort and quit. Please turn in your code, but do not turn in the people.txt, people\_sorted.txt, or people\_out.txt data files.

If desired, temporarily save this part to the file: **DDHH\_L7\_Lastname\_Menu.py**

**Summary: The following files are created in this lab:**

**DDHH\_L7\_Lastname\_Sort.py**

**DDHH\_L7\_Lastname\_Menu.py**

**Extra Credit:** Add options: u)ppercase and l)lowercase which will uppercase or lowercase all the names in the list.

When finished, role all your work into one file and submit: **DDHH\_L7\_Lastname.py**