Advanced Topics Assignment: Students
Reading in from a file, a List of Tuples, Searching and Sorting

What is a tuple?

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list. A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

Advantages of Tuple over List in Python

Since tuples are quite similar to lists, both of them are used in similar situations.
However, there are certain advantages of implementing a tuple over a list:

- We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.
- Since tuples are immutable, iterating through a tuple is faster than with a list. So there is a slight performance boost.
- Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

Assignment:

Stage 1
Read in the data from a file into a **list of tuples**. Each line in the data file represents one student's information (first name, last name, average). Each line must be stored in its own tuple, therefore each tuple represents one student's information. All the tuples are stored in a list. The information can be found in a file that has been provided called Student Marks. I have provided sample code below for reference.

Stage 2
Pick the sorting algorithm that makes the most sense to you (Bubble, Selection or Insertion). Give the user the choice to sort by either last name or by average. Remember to swap the entire tuple and not just one element. Display the results (first name, last name and average must all be displayed).

Stage 3
Using the Binary Search Algorithm. Allow the user to search for a student by last name. Once found, the student's entire name (first and last) and average should be displayed to the user. (Please note that the Binary Search algorithm can only be applied to a sorted list. Since the search is by last name the sort by last name must be called prior to this.)

On the following pages, I have provided some code that may be helpful.

Reading the file and creating a list of student tuples (first name, last name, mark)

```
def readFile(file):
    """
    this function reads the contents of a file line by line and stores them in a list of tuples
    :param file: the file to be opened and read
    :return: the filled list
    """
    i = 0
    tupleList = []    #the list for filling with the file contents (square brackets for lists)
    with open(file) as f:   #opens the file
        while True:  #loops for each line of the file
            line = f.readline()  #reads in each line from the text file
            if not line:   #no more lines (end of file was found)
                break     #exits loop when the eof (end of file) marker is found
            tmp = line.split(" ") #splits the line according to whitespace found
            tupleList.append((tmp[0], tmp[1], float(tmp[2]))) #adds next tuple to the list (round brackets
indicate tuples)
        f.close()  #closes the open file
        return tupleList # returns the filled list to the main program


def printContents(studentList):

    """
    this function print the contents of a list
    :param studentList: the list of student tuples to be displayed
    """
    for i in range(0,len(studentList)):  #for the length of the list
        for j in range (0, 3):
            print(studentList[i][j], end = " ") #outputs 3 pieces of student data on same line
        print(" ") #new line

def main():
    studentInfo=[]; #list for the contents of the file
    studentInfo = readFile("H:\ICS 4U - ND\Student Marks.txt") #fills studentInfo with a call to
readFile
     printContents(studentInfo) #outputs studentInfo with a call to printContents

main() #call to main
```

Iterative Binary Search

Note that this binary search is on an integer list. You have a list of tuples. You are sorting by an element of the tuple, either the last name or the average according to the user's preference.

```
# It returns location of x in given array arr
# if present, else returns -1


def binarySearch(arr, min, max, x):

    while min <= max:

        mid = min + (max - min) // 2

        # Check if x is present at mid
        if arr[mid] == x:
            return mid

        # If x is greater, ignore left half
        elif arr[mid] < x:
            min = mid + 1

        # If x is smaller, ignore right half
        else:
            max = mid - 1

    # If we reach here, then the element was not present
    return -1


# Driver Code
list = [2, 3, 4, 10, 40]
num = int(input("What number would you like to search for?"))

# Function call
result = binarySearch(list, 0, len(list)-1, num)

if result != -1:
    print("Element is present at index % d" % result)
else:
    print("Element is not present in list")
```

Students

STRAND		Marks
Programming Skills and Concepts / 27	• Header contains name, date, description	/1
	• Proper format (indentation, end each structure (ie loops, if, methods, classes)	/1
	• All variables commented	/1
	• Methods properly commented with param and returns described	/3
	• Class structure defined properly	/1
	• Array of Student Tuples declared and filled	/3
	• Sort algorithm coded properly providing the desired results when sorting by last name	/5
	• Sort algorithm coded properly providing the desired results when sorting by average	/5
	• Binary Search algorithm coded properly providing the desired result	/5
	• Error message if desired item not found	/2
Designing Modular Programs /10	• Proper use of methods (main basically calls methods) • Inclusion of a class	/10

Software Development /25	· List of all main and class variables with brief description · Method headers of each method and brief description · Algorithm or Flow chart of main method ·Flow Chart or pseudocode of your Chosen Sort Algorithm ·Flow Chart or pseudocode of the Binary Search Algorithm	/ 25 (5 marks each)