**COSC 1336: Fall 2022**

# Assignment for Chapter 11

**DUE: November 22, 2022, by 1 AM**

**20% PENALTY PER EACH DAY**

**[50 points] Question 1**

The following refer to a file similar to "qbdata.txt" in section 11.3 of the Runestone book^. You need to write the following functions:

**QB_search(filename, searchterm)** that takes a file name and a string **searchterm** as parameters and returns a list of QB `firstnames` in the corresponding file, where the last name or part of the `lastname` matches **searchterm.**

- `QB_search(filename, 'Manning')` will return `['Payton', 'Eli']`
- `QB_search(filename, 'McCoy')` will return `['Colt']`
- `QB_search(filename, 'Patton')` will return `[]`
- `QB_search(filename,'an')` will return `['Josh', 'Peyton', 'Matt', 'Mark', 'Eli']`

**QB_great(filename)** that takes a file name as a parameter and returns a list of quarterback `lastnames` with more than 3 times as many touchdowns as interceptions

- `QB_great(filename)` will return `['Freeman', 'Vick', 'Ryan', 'Cassel', 'Brady', 'Roethlisberger']`

**QB_legend(filename)** that takes a file name as a parameter and returns a list of quarterback `lastnames` with more than (1) 3 times as many touchdowns as interceptions, (2) 60% QB ratings and (3) 30 touchdowns

- `QB_legend(filename)` will return `['Brady']`

Example:

```
myFile = "qbdata.txt"

print(QB_search(myFile, 'Manning'))
print(QB_search(myFile, 'McCoy'))
print(QB_search(myFile, 'Patton'))
print(QB_search(myFile, 'an'))
print(QB_great(myFile))
print(f"{QB_legend(myFile)[0]} is the legendary QB!")
```

Output:

```
['Payton', 'Eli']
['Colt']
[]
```

```
['Josh', 'Peyton', 'Matt', 'Mark', 'Eli']
['Freeman', 'Vick', 'Ryan', 'Cassel', 'Brady', 'Roethlisberger']
Brady is the legendary QB!
```

You can test the program by reading the file in your main program and calling these functions.

^ Each line has data in the following order: First name, Last name, Position (QB), Team, Attempts, Completed, Touch Downs, Interceptions, Completion Rate (Attempts/Completed), QB Rating.

So, for the first line, we have: Colt McCoy QB CLE 135 222 1576 6 9 60.8% 74.5

We can interpret this line as below:

First Name: Colt

Last Name: McCoy

Position: QB

Team: CLE

Passes Attempted: 135

Passes Completed: 222

Total Pass Yards: 1576

Touchdowns: 6

Interceptions: 9

Pass Completion Rate: 60.8%

QB Rating: 74.5

**[50 points] Question 2**

You need to write a set of functions as follows. Two example files are provided for testing: **`JackAndJill.txt`** and **`TheHillWeClimb.txt.`**

**`file_basics(inputFilename)`** that takes a filename (string) as a parameter and returns a list containing the following in order: i) Number of characters and ii) Number of words in the file. You can assume words are separated by blanks.

- `file_basics("JackandJill.txt")` will return the list `[125, 25]`
- `file_basics("TheHillWeClimb.txt")` will return the list `[3862, 710]`

**`match_words(inputFilename, outputFilename, Wordlist)`** that takes **`inputFilename`** (`string`) and **`Wordlist`**, a list of words as parameters and creates a file (with **`outputFilename`**) containing the number of times each word in the wordlist appears in the input text file.

Cases

```
wlist1 = ["Jack", "Jill", "up", "river"]
match_words("JackandJill.txt", "outfile1.txt", wlist1)
```

will result in a file outfile1.txt with the following content:

```
Jack 2
Jill 2
```

```
up 1
river 0

wlist2 = ['up', 'rise', 'we', 'nation']
match_words("TheHillWeClimb.txt", "outfile2.txt", wlist2)
```
will result in a file outfile2.txt with the following content:
```
up 1
rise 4
we 38
nation 3
```

Example:
```
print(file_basics("JackandJill.txt"))
print(file_basics("TheHillWeClimb.txt"))

wlist1 = ["Jack", "Jill", "up", "river"]
match_words("JackandJill.txt", "outfile1.txt", wlist1)

wlist2 = ['up', 'rise', 'we', 'nation']
match_words("TheHillWeClimb.txt", "outfile2.txt", wlist2)
```

Output:
```
[125, 25]
[3862, 710]
>> Creates "outfile1.txt"
>> Creates "outfile2.txt"
```

**[10 points] Question 3 – Extra Credit Question**

Write a program that reads a file named 'studentdata.txt' that contains the names of students in a section of COSC 1336 along with their exam scores separated by spaces. The first four entries in the file contain scores out of 100 points each as follows**: [classwork homework midterm final exam]**. Any additional entries can be ignored. Your program should print each student's cumulative score. The classwork is 10%, homework is 40%, midterm is 20%, and the final exam is 30% of the cumulative score

Example of data in the file 'studentdata.txt':
```
joe 10 15 20 30 40
bill 23 16 19 22
sue 8 22 17 14 32 17 24 21 2 9 11 17
grace 12 28 21 45 26 10
john 14 32 25 16 89
```

Your program output should look like the example below for the given file (studentdata.txt):

**joe has a cumulative score of 20.00**
**bill has a cumulative score of 19.10**
**sue has a cumulative score of 17.20**
**grace has a cumulative score of 30.10**
**john has a cumulative score of 24.00**