Homework #1 Data Structures Due: February 2 (Saturday at 11:59 PM)

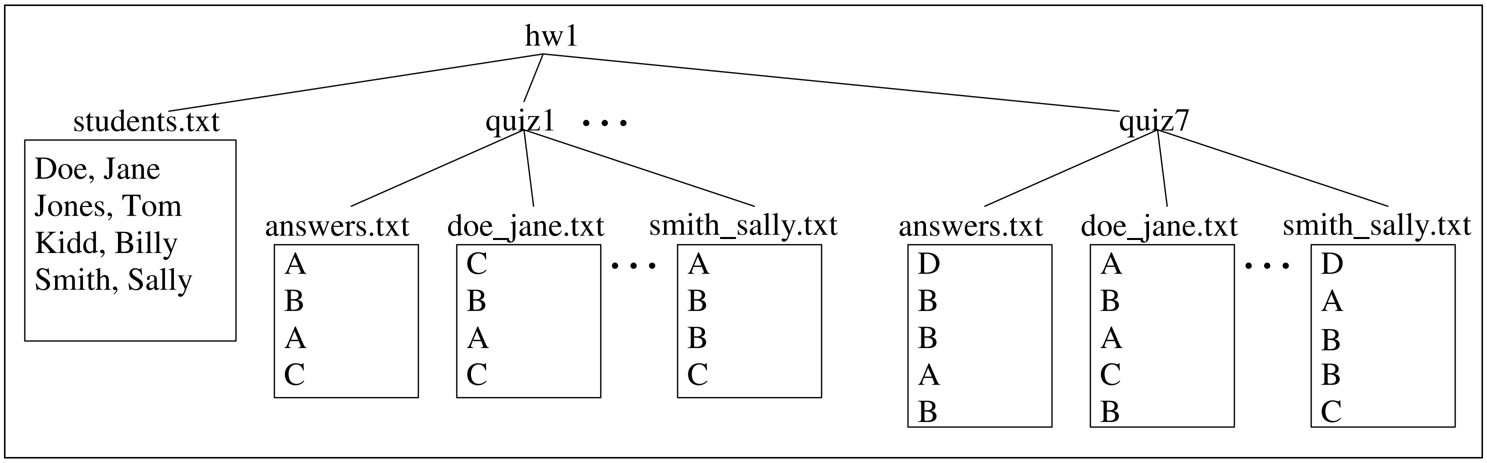
Objects: Practice designing a program and "reviewing" Python file and `os` module usage. (NOTE: be sure to review the **example programs** `formattedOutput.py` **and** `changeDirectory.py` **found at:** www.cs.uni.edu/~fienup/cs1520s19/homework/example_programs_hw1.zip)

Electronic Quiz Grader Program

The eLearning multiple-choice-quiz grader has broken down, so Professor Smart N. Lazy wants you to write a program (`quizGrader.py`) to grade the class's eLearning quizzes. After extracting the files from:
http://www.cs.uni.edu/~fienup/cs1520s19/homework/hw1.zip
you will find that the hw1 folder contains:

- `students.txt` - a text file containing the student names in the class
- one **or more** `quiz#` directories - each directory contains an `answers.txt` text file with the correct answers and text files for each student who took the quiz. The student file names are `lastname_firstname.txt`

WARNING: don't tailor your program to only work with these students (e.g., Doe, Jane) or quizzes. You program should work with different data files than are named the same (e.g., there is always a `students.txt` file, etc.)



Your program (called `quizGrader.py`) should run from inside the hw1 directory (i.e., develop it inside the hw1 directory) to generate a `gradeReport.txt` file that looks something like:

```
         Student Quiz Report

                  Total       Overall
Student         Quiz Points    Quiz %
-------------------------------------------
Doe, Jane           30          71.4
Jones, Tom          40          95.2
Kidd, Billy         35          83.3
Smith, Sally        36          85.7

Points Possible     42
```

For extra credit, you can report more details (e.g., individual quiz scores for each student):

```
                         Student Quiz Report

                                                    Total     Overall
Student             Quiz 1  Quiz 2       Quiz 7  Quiz Points  Quiz %
--------------------------------- ... -------------------------------
Doe, Jane              3       5            2         30        71.4
Jones, Tom             4       6            5         40        95.2
Kidd, Billy            3       4            4         35        83.3
Smith, Sally           3       6            1         36        85.7

Points Possible        4       7            5         42
```

When you write your program, be sure you:

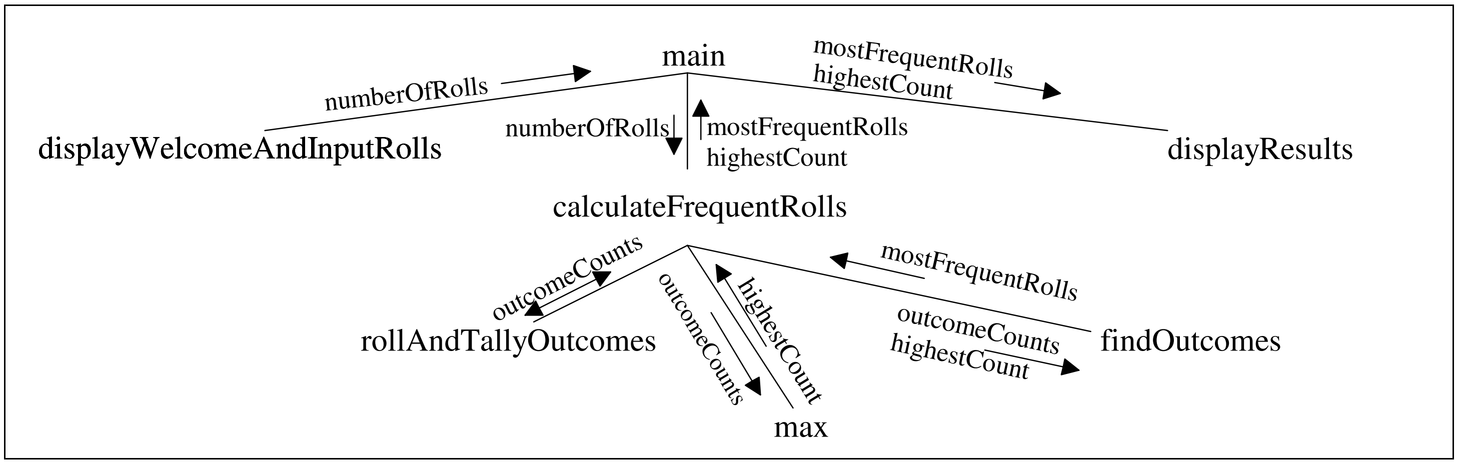
- think about the functional-decomposition (top-down) design before you start to write code. You'll need to turn in a design document, so you should might as well start there. Hint: model what a teach would do by hand.
- think about built-in Python data structures (lists or dictionaries)
- use meaningful variable names with good style (i.e., useCamelCase)
- use comments (""" Multi-line Comment """) at the start of the program **and** immediately after each function `def`inition describing what they do (see lab1 `diceOutcomes.py` program)
- use a main function (see lab1 `diceOutcomes.py` program) located at the top of program with a call to it at the bottom to start execution
- use global constants where appropriate with good style (ALL_CAPS_AND_UNDERSCORES). (Put your global constants after your initial comments describing the program and before your main function definition so they can be found and changed easily in future versions of your program.)

Submit your homework electronically at https://www.cs.uni.edu/~schafer/submit/which_course.cgi

Submit a single zipped file, hw1.zip containing the following:

- **`quizGrader.py`** (your Python program)
- **`design.doc`** (or design.pdf, or design.txt, design.jpg, or design.rtf) a document describing the design of your program including a functional-decomposition diagram with text describing each function. The level of detail should be similar to **lab1 description** included below!
- original data files and directories contained in hw1.zip downloaded (students.txt, etc.)

Recall the Lab 1 functional-decomposition that you can model for this homework's design document. Your design diagram should include the flow of information between your functions.



For each function, include a brief description of what it does (NOT the code/algorithm):

`main` - provides an outline of program by calling top-level functions

`displayWelcomeAndInputRolls` - Displays welcome message for the user. Gets and returns the number of dice rolls from the user.

`calculateFrequentRolls` - Rolls the dice the correct number of times, tallies the outcomes, and returns a list of outcomes with the highest count and highest count.

`rollAndTallyOutcomes` - Rolls the dice the correct number of times and tallies the outcomes. Returns a list of tallies with the index being the outcome.

`max` - built-in function to return the largest item in an iterable data structure like a list.

`findOutcomes` - Returns a list of outcomes with the highest count.

`displayResults` - Displays the outcome(s) with the highest percentage.