COP 2930 - Individual Programming Assignment #9

Due date: Please consult WebCourses for your due date/time

Objectives

1. Practice writing functions that processes multiple lists.
2. Practice with simulating a process described by list input.

Problem A: Electoral Vote Function (electoralvote.py)

For the purposes of this problem, the Electoral College works as follows:
Only two candidates run for the office of president.
Each state is assigned a number of electoral votes.
Whichever candidate has more votes in a state win ***all*** of its electoral votes.
Whichever candidate earns the most electoral votes wins the election.

Write a function that takes in the three following lists, all guaranteed to be the same length:

1. votesA
2. votesB
3. electoralVotes

Each index in each of these lists stands for a particular state. At each index, votesA stores the number of votes candidate A received for that state. At each index, votesB stores the number of votes candidate B received for that state. At each index, electoralVotes stores the number of electoral votes that state is worth.

Write a function, which, after taking in this information, calculates the number of electoral votes earned by candidate A and returns this value.

```
def countElectoralVotes(votesA,votesB,electoralVotes):
    # Fill in your code here.
```

This time, please write your own test cases and test function. Here is one test case:

votesA = [100, 50, 80, 99, 210]
votesB = [85, 51, 79, 3, 211]
electoralVotes = [10, 2, 20, 17, 3]

In this case, candidate A wins states 0, 2 and 3, earning 10, 20 and 17 electoral votes, respectively, for a total of 47 electoral votes.

Problem B: When Will the Voting Booth Close (closing.py)

Each polling location has several private booths for voting. Since the ballot is so long these days, most citizens take several minutes to vote. One system a polling location could use is to always rotate through its booths, so that the first person in line goes to booth 1, the second person in line goes to booth 2, etc. when a person gets assigned to the last booth, the next person in line goes to booth 1, when the person in booth 1 is finished.

It's almost 7 pm and the poll workers are very tired. They want to go home!

Unfortunately, as long as someone arrives to the polls by 7 pm, they must be allowed to vote.

For this problem, you will write a function that calculates how many minutes after 7 pm the poll workers will have to stay after the last person has completed voting.

The function you will write will take in two inputs:

1. voteTimes, a list of how long each person currently in line will take while they are in the private booth to complete voting.

2. numBooths, an integer representing the number of polling booths at the precinct.

Given these inputs, your function should return an integer representing how many minutes past 7 pm the voting will complete.

Consider the following example:

```
voteTimes = [3, 8, 2, 9, 6, 5, 12, 3, 4, 4, 8, 9, 7]
numBooths = 5
```

Let's number the booths 0, 1, 2, 3 and 4. (This is how it will be in code.)

In this example, the first five people get to the first five booths immediately, at 7 pm, taking 3, 8, 2, 9 and 6 minutes respectively.

The person who is sixth, will be assigned to booth 0. He waits 3 minutes until the previous person in booth 1 finishes, and then he'll start voting.

The person who is seventh with start voting at 7:08 pm.
The person who is eighth will start voting at 7:02 pm. (Note that the system isn't perfectly fair or efficient…even though this person arrived later, they get to vote earlier. The reason for this rule in the problem is simply just to make it easier for you to code.)
The person who is ninth will start voting at 7:09 pm. This person, who takes 4 minutes to vote, will finish at 7:13 pm, and will be the last person to vote in booth #3.

The person who is tenth will start voting at 7:06 pm. This person, who takes 4 minutes to vote, will finish at 7:10 pm, and will be the last person to vote in booth #4.
The person who is 11th will start voting at 7:08 and will finish at 7:16 pm, at which point booth #0 will be clear.
The person who is 12th will start voting at 7:20 pm (behind people who took 8 and 12 minutes, respectively), and finish at 7:29 pm, at which point booth #1 will be clear.
The person who is 13th will start voting at 7:05 pm and finish at 7:12 pm, at which point booth #2 will be clear.

Thus, the last person completes voting at 7:29 pm, so the poll workers have to wait 29 minutes past 7 pm.

```
def timeToWait(voteTimes,numBooths):
    # Fill in your code here.
```

This time, please write your own test cases and test function. A good starting point would be to use the test case described above.

Restrictions

Please IDLE 3.6 (or higher) to develop your programs. Write each in a separate file with the names specified previously, **electoralvote.py** and **closing.py**

Each of your **two** programs should include a header comment with the following information: your name, course number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem. Your code for each part should have both your function as well as at least two test cases in a separate testing function that you create. **Note: We will still test your function on our test cases though.**

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness and filling in the given function prototype appropriately.

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility to IDLE.