Junior Knights – List and File Practice

Objectives

1. Learn how to read input from a file.
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.

File Input Format (electoralvote.in)

The first line of the input file will have a single integer, ***e***, representing the number of elections to process. (Note: please do NOT call your variable name that reads this in e.)

The fine line of input for each election will have ***n*** space separated integers, representing the number of votes candidate A got in each of the states, in order. (First number is state 0, second number is state 1, and so forth. Note: ***n*** is the number of states for this election.)

The second line of input for each election will have ***n*** space separated integers, representing the number of votes candidate B got in each of the states, in order.

The third line of input for each election will ***n*** space separated integers, representing the number of electoral votes for each state, in order.

In particular, it's guaranteed that all three lines of input for each election will have the same number of positive integers on them, representing the number of states for that input case.

It is guaranteed that there will NOT be a tie for each state.

Output Format

For each election, print out a single line with two space separated integers: the number of electoral votes earned by candidate A and the number of electoral votes earned by candidate B.

Required Functions

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.
```

Sample Input File

```
2
5
100 50 80 99 210
85 51 79 3 211
10 2 20 17 3
8
1 2 3 4 5 6 7 8
2 3 4 5 6 7 8 9
100 200 300 400 500 600 700 800
```

Sample Output

```
47 5
0 3600
```

Problem B: When Will the Voting Booth Close (closing1.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.

File Input Format (closing1.in)

The first line of the input file will have a single integer, ***p***, representing the number of polling locations to process. (Note: please do NOT call your variable name that reads this in p.)

The first line of input for each polling location will have a single positive integer, ***b***, representing the number of booths at the polling location.

The second line of input for each polling location will have a list of positive integers, in order, representing the number of minutes each person in line at 7 pm will take to vote once they arrive at their respective voting booth.

Output Format

Output a single integer representing the number of minutes after 7 pm all of the voting will complete.

Required Function

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. Here is the function prototype:

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

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.

Sample Input File

```
2
5
3 8 2 9 6 5 12 3 4 4 8 9 7
7
1 2 3 4 5 6 19 1 16 15 14 13 1
```

Sample Output

```
29
19
```

Problem C: When Will the Voting Booth Close (closing1.py)

After the contentious 2020 election, several polling locations realized that their old way of assigning voting booths to people waiting in line was inefficient. For example, if there are five booths, and the first person takes 10 minutes to vote while persons 2 through 5 only take 2 minutes to vote, the sixth person in line has to wait for 10 minutes even though there are other open lines. A better policy would be to assign the sixth person to the line that is free the earliest. Solve this problem using this strategy instead. In this solution, keep all of our code from your solution to problem B, **except** the function timeToWait. Please rewrite this function to simulate this new, improved voting process! The input file upon which this program will be tested will be the exact same as the one for Problem B.

Sample Input File (closing.in)

```
2
5
3 8 2 9 6 5 12 3 4 4 8 9 7
7
1 2 3 4 5 6 19 1 16 15 14 13 1
```

Sample Output

```
19
19
```

Here is how the first sample input works differently with this new voting procedure.

```
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 2. He waits 2 minutes until the previous person in booth 1 finishes, and then he'll start voting, and will finish at 7:07.
The person who is seventh with start voting at 7:03 at booth 0 and will finish at 7:15.
The person who is eighth will start voting at 7:06 at booth 4 and will finish at 7:09.
The person who is ninth will start voting at 7:07 at booth 2 and will finish at 7:11.
The person who is tenth will start voting at 7:08 at booth 1 and will finish at 7:12.
The person who is 11^{th} will start voting at 7:09 at booth 3 and will finish at 7:17.
The person who is 12^{th} will start voting at 7:09 at booth 4 and will finish at 7:18 pm.
The person who is 13^{th} will start voting at 7:12 at booth 1 and finish at 7:19 pm.

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

FYI: As the complete test input shows, this isn't always ideal. For the third to last case, it's actually slower than the procedure in program B!