UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

Assignment 7

Arrays

Date Due: Thursday Oct. 29, 2020, 11:59pm **Total Marks: 13**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work. **Put your name, NSID, student number and instructor's name at the top of the document.** This assists the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.
- Programs must be written in Python 3.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Questions are annotated use descriptors like "easy" "moderate" and "tricky". All students should be able to obtain perfect grades on "easy" problems. Most students should obtain perfect grades on "moderate" problems. The problems marked "tricky" may require significantly more time, and only the top students should expect to get perfect grades on these. We use these annotations to help students be aware of the differences, and also to help students allocate their time wisely. Partial credit will be given as appropriate, so hand in all attempts.

Question 1 (5 points):

Purpose: To practice working with 2D arrays.

Degree of Difficulty: Easy.

Officer Jenny is in charge of training new police recruits. She just gave her students a multipe-choice quiz. She wants to produce a summary of student performance for all questions where **fewer than 60% of the recruits got the question right**. Write a program to accomplish this task.

Using Numpy Arrays

Although in principle this question could be solved using lists only, our goal is to gain practice working with arrays. Therefore, you must convert the quiz data to an array and perform all calculations using array operations. Before you start anything, make sure you have imported the `numpy` module. This module is NOT standard, so if you are working on your own machine, you may need to install it first.

Quiz Data

A starter file is provided for you that contains the quiz data as a list-of-lists. The data is organized such that each sublist represents the quiz result for a single student. Each sublist is the same length and contains multiple integers, indicating the student's score on each question on the quiz. A value of 1 means the student got the question right and 0 means the student got the question wrong.

The program you hand in should compute the result for the `quizResults` list, but two other very small lists are provided to help you perform simple testing. For example, `test1` represents a quiz with just 2 questions that was written by 3 students, where all 3 students got the first question wrong.

Program Behaviour

Your program should:

(a) Create a 2D array from the list of lists for a quiz. Your program should work no matter how many students there are, or how many questions in the quiz.

(b) For each question, calculate the percentage of students who answered incorrectly

- If fewer than 60% answered correctly, print the information (performance and question number) for that question to the console

Take advantage of the power of arrays to do this! In particular, you can slice a 2D array across multiple dimensions; think about how this might be useful.

Sample Run

```
Poorly done questions:
Only 45 percent of students solved question 1
Only 55 percent of students solved question 3
Only 50 percent of students solved question 8
```

What to Hand In

(a) A document entitled `a7q1.py` containing your finished program, as described above.

Evaluation

- 1 mark for creating a 2D array
- 3 marks for calculating the student performance for each question
- 1 mark for correct console output
- -1 mark if the student did not include their name, NSID, student number and instructor's name at the top of the submitted file.

Question 2 (8 points):

Purpose: To practice simple file I/O and numpy array operations

Degree of Difficulty: Moderate.

For provincial election purposes, the province of Saskatchewan is divided into constituencies. Each constituency elects a representative to send to the provincial parliament. Nearly always, each representative belongs to a registered political party.

In 2016, the Saskatchewan Party won the election with a total of 51 out of 61 constituencies (you can see the full results here: `https://results.elections.sk.ca/ge28/`). For this question, you will use real election data to determine how many of those wins were a **majority victory**, i.e. where the Saskatchewan Party won more than half of the votes cast in the constituency.

Using Numpy Arrays

The main purpose of this question is to practice data manipulation using numpy arrays.

You will need all three of: array arithmetic, array relations, and logical array indexing. You should be able to use all three of these techniques in a useful way while solving this problem.

Before you start anything, make sure you have imported the `numpy` module. This module is NOT standard, so if you are working on your own machine, you may need to install it first.

Task Breakdown

The steps below will help you break down this process into manageable pieces.

Load the Data into Arrays

Download the election results data file from the course website. It looks like this:

```
constituency,green,ndp,pc,liberal,saskparty,wip
ARM RIVER,241,1457,338,207,6187,0
ATHABASCA,53,1756,0,262,644,0
...
```

The first line of the file is a header that lists all the political parties. The other lines list the name of a constituency and the number of votes won by each party (in the order shown in the header) in that constituency.

Write code to open this file and read the election data into **two separate lists**: one for the constituency names and one for the vote tallies. Then, convert each list to a **numpy array**. Make sure to maintain the order so that the arrays line up by position (i.e. `ARM RIVER` is first in the array of constituencies, and its matching votes are first in the array of votes)

Test these arrays before you move on! From this point on, you must perform all calculations using arrays rather than lists.

Total Votes by Constituency

Next, create an array that contains the total number of votes cast in each constituency, again making sure to maintain the order of the data. Take good advantage of array operations such as slicing and array arithmetic here.

Finding the Majority Wins

Finally, use the arrays you have created so far to find all of the constituencies where the Sask Party won more than half of the total votes cast in that constituency. Make good use of **array relations** and **logical indexing** here. In particular, you can use **array relations** on two arrays so long as they are the same size; this might be useful to do using the array of total votes that you created in the previous step.

Note that you **cannot assume** what the index of the Sask Party's vote column will be. Use the header info from the file to find it!

Your program should print to the console the **number** of constituencies where the Sask Party won a majority victory, as well as the names of those constituencies. Nice formatting is not required: simply printing all of the constituency names in an array is fine.

Sample Run

The output of your program might look something like this:

```
The Saskatchewan Party won a vote majority in 45 constituencies.
---------
['ARM RIVER' 'BATOCHE' 'BIGGAR-SASK VALLEY' 'CANNINGTON' 'CANORA-PELLY'
...
...
 'WEYBURN-BIG MUDDY' 'WOOD RIVER' 'YORKTON']
```

What to Hand In

(a) A document entitled `a7q2.py` containing your finished program, as described above.

Evaluation

- 2 marks for loading the data into numpy arrays
- 2 marks for making good use of array slicing and arithmetic to find the total votes
- 3 marks for making good use of array relations and logical indexing to find the majority wins
- 1 mark for correct console output
- -1 mark if the student did not include their name, NSID, student number and instructor's name at the top of the submitted file.