Assignment 7

Arrays

Date Due: 5:00pm Friday Nov 8 **Total Marks: 15**

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.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M.
- Make sure your name and student number appear at the top of every document you hand in. These conventions assist 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, or zip files, even if you think it will help. It might help you, but it adds an extra step for the markers.
- Programs must be written in Python 3.5+
- You may not, in part or in whole, submit any code that was written by generative AI tools, such as ChatGPT
- **Assignments must be submitted to Canvas (the course website).**
- **Canvas will not let you submit work after the assignment hand-in closes.** 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.
- Read the purpose of each question. Read the Evaluation section of each question.

Question 1 (10 points):

Purpose: To practice string and file manipulation; to solve a problem using numpy

Solving systems of (linear) equations is a common task in mathematics. For example, here is a small system of three equations with three unknowns:

```
2x + y + 3z = 9
5x + 3y + z = 15
x + 7y + 12z = 0
```

Solving the system means finding numeric values for all the unknowns, in this case, x, y, and z, such that all three equations are simultaneously true.

Small systems of equations can be solved by hand, but most programming languages, and certainly Python, have libraries that can solve them for you. You'll explore that idea in this question.

Using Numpy

The programming parts of this assignment will require using the `numpy` module. This module is NOT standard, so if you are working on your own machine, you may need to install it first. Depending on how you installed Python, this can hopefully be as easy as pulling up a terminal and typing `conda install numpy`.

Part 1: Hand Solve

First solve the system of equations found in the provided file `system1.txt` (it's the same one as the example shown just above). Do it by hand using any method you like, but without using a program of any kind. Show your work and hand it in. You can round any decimal numbers in your final answer to 4 figures of precision.

Part 2: Solve using numpy

To verify your work from part 1, write a very small program that uses the numpy module to solve the system of equations from `system1.txt`. The method you need from the numpy module is `numpy.linsolve.solve()`. The only trick comes from figuring out the format that you need to use for your system in order to 'feed it in' to the `solve()` method.

In order to do that, go ahead and use ChatGPT! Go to ChatGPT (or any other large language model that you like) and ask it something like "show me a simple example of how to solve systems of linear equations in Python". Don't tell ChatGPT anything about your SPECIFIC system of equations. Just use the example it gives you to write your own program that solves that system. Most likely your program will be no more than 5 lines of code. Check it against the answer you figured out by hand, and then hand this program in.

Part 3: Write an equation parser

Take a look at the systems of equations found in `system2.txt` and `system3.txt`. Would you want to solve these systems by hand or even manually plug them in to the format that `numpy.linsolve.solve()` requires? If you said 'no', then good. You're ready for the main part of this question.

Your last task is to write a program that:

- asks the user to enter the name of a file containing a system of equations
- extracts the necessary info from that file and solves the system using numpy
- prints the solution to the console in a simple but readable way

Equation file format

The systems of equations will be given in files similar to the provided examples. You can count on the following simplifying guarantees concerning that file format:

- Each line of the file will be a single equation
- All of the variables and their coefficients will be on the left hand of the equal sign
- All of the variables in each equation will be the same, and in the same order
- Every equation will include every variable (using a coefficient of 0 if needed)
- All of the constants and coefficients will be integers (not floats)
- The variables will be joined exclusively with + signs (the coefficients themselves may be negative)
- The first character of each variable will always be an alphabetic letter

You will find the provided equation files match these rules, and your program should work on any other file that also follows these rules.

Problem Decomposition

Write **3 separate functions** that will divide the overall work for this problem, and that when called in sequence, will extract the info you need from the equations file in the format required by numpy. We'll describe those functions here.

Note that this is NOT the only way to solve this problem; it's exposing you to just ONE reasonable way to decompose this problem into manageable chunks.

Function 1: Get the equations

Write a function that accepts the **file name** where the equations are scored, and returns TWO separate lists. The first list should be a list of strings, where each string is one of the equations (i.e. everything up to the equal sign). The second list should be a list of integers of all the constants (i.e. all the numbers on the right hand sign of the equal sign). For example, for `system1.txt`, the lists returned from this function should look like this:

```
['2x + y + 3z', '5x + 3y + z', 'x + 7y + 12z']
[9, 15, 0]
```

Function 2: Get the variable names

Write a function that accepts a **list of equations** (like the list produced by function 1) and returns a list of all the variable names from the system (you'll need this simply to print out the solution a little more nicely at the end). For example, for `system1.txt`, the list returned should look like this:

```
['x', 'y', 'z']
```

Function 3: Get the coefficients

Write a function that accepts a **list of equations** (like the list produced by function 1) and returns a list-of-lists of integers, where each sublist contains all the coefficients of a SINGLE equation. For example, for `system1.txt`, the list returned should look like this:

```
[[2, 1, 3], [5, 3, 1], [1, 7, 12]]
```

Main Program

If all three of your functions are working, the main part of your program will be very simple. It only needs to ask the user for the file name, call the three functions in sequence, feed the resulting lists into the linear solver from numpy, and then use a simple loop of some kind to nicely print out the solution to the system.

Sample Run

The overall output of your program might look something like this (except we've replaced the actual solution with question marks, so as not to spoil the answer that you're supposed to compute by hand!):

```
Enter equation system file name for solving: system1.txt
Solution for equations from system1.txt
---------
x = ?.????
y = ?.????
z = ?.????
```

What to Hand In

(a) A document entitled `a7q1_pt1.txt` (or .pdf or .png) showing your hand solution for part 1
(b) A python program `a7q1_pt2.py` with your tiny program for part 2
(c) A python program `a7q1.py` containing your program for part 3

Evaluation

- 3 marks for a correct and readable hand solution for part 1
- 1 mark for the simple hard-coded solution from part 2
- 3 marks for correctness of results from part 3
- 3 marks for function design matching the specification

Question 2 (3 points):

All programs that you submit for this assignment will be assessed with regard to their **style and readability**. Make sure that all of your code:

- includes your name, nsid, and student number as comments
- uses informative variable names
- makes effective use of formatting and white space
- includes a `docstring` for all **function definitions** in the program
- for longer programs, uses helpful and concise single-line comments where appropriate

What to Hand In

There is nothing to hand in for this question. It's simply describing how the rest of your work will be evaluated with regard to style and readability. It is possible to earn these marks even if you don't hand in every other question in this assignment, but you must hand in enough work such that your style and readability can be adequately assessed.

Evaluation

- 3 marks for consistently following the principles listed above

Question 3 (2 points):

In your submission for this assignment, you have an opportunity to earn something we'll call "the wow factor". To earn the wow factor, you must do something creative that goes beyond the expectations laid out in the assignment. This could be an extra feature for one of the previous questions, or it could be an extra little program that you submit that builds on the skills used in the assignment.

You should not aim to do a lot of work for this, but should instead demonstrate in a simple and effective way the depth of your understanding. In short, take ownership of your work and "impress us".

Telling you exactly what to do to earn "the wow factor" would defeat the entire purpose, but here are some very general ideas.

- improve the aesthetic appeal of your program(s)
- improve the user-friendliness of your program(s)
- remove a simplifying assumption and handle the resulting added complexity
- compare two different ways of doing the same task, and submitting data/analysis on which was better
- just doing something generally cool

One quick warning: using fancy extra code libraries in order to AVOID a key learning objective from the assignment is not impressive, it's actually anti-impressive. Focus instead on solving problems using simple clean foundational programming skills. Using extra libraries is fine if they let you do something BEYOND what the assignment asked, but not if they're replacing a core assigned task.

What to Hand In

A plain text file called `wow.txt` describing what you did that you think merits the wow factor. One or two sentences is enough. If you create any additional code files, hand those in too.

Evaluation

- 2 marks for something impressive