EXAM PYTHON (100 POINTS)

- This is an examination. It is to be your work and your work alone.
- No exchange of information with another human entity in any form is acceptable.
- It is ok to read Google documents, however no exchanges via the internet are acceptable.
- It is ok to use class material, notes, your programs, your labs, and any other notes you have written for class.
- It is ok to use a textbook.

These instructions are general for all the scripts that you will write in python3.

```
Instructions
#class header

#Q1
var1=124
print(var1)

#Q2
var2=3+var1
print(var2)
```

```
Deadline and late penalty
The deadline is 8:30pm with 3 min grace period.
After 8:33pm, the penalty is 2 points for each late minute.
Gradescope Exam2 will close at 8:40m

IF YOU HAVE TECHNICAL ISSUES DURING THE EXAM LET THE PROFESSOR KNOW
```

Policy for cheating: sharing code on the exam is unacceptable and will earn you a 0 and take you straight to the ethics board. Do not share any study documents either!

We give partial points in case of syntax errors.

We **will subtract -0.5 points for missing or incorrect header and/or Q&A for each script.**

After you download the Exam2-S2023.zip from Canvas, unzip the file if necessary, and a directory called **Exam2-S2023** will be created. Change into Exam2-S2023 and start your work in that directory.

We take points off for using python concepts or modules we have not discussed in class.

1. **(20)** The provided script **ex1-s23.py** contains the following dictionary

```
D={'s1': ['g','a','t','*'], 's2':['t','a','c','&'], 's3':['a','%']}
```

Q1. (10) Your first task is to remove the last character (symbol) from each of the values of **D**, and to build a list called **Lchar** of the removed characters. Use one for loop or one list comprehension. **You should loop over the values of D.**

Within the for loop or list comprehension, use a list method that removes an item and returns that item.
Print the modified dictionary D, and list Lchar to screen.

```
{'s1': ['g', 'a', 't'], 's2': ['t', 'a', 'c'], 's3': ['a']}
['*', '&', '%']
```

Q2. (10) Use one for loop over the values of the modified dictionary **D** to build this string

```
GATTACA
```

Within the for loop, you should use a string method to convert a list of strings to a string, and another string method to make that string uppercase. The final string should be built within the loop.
Print the final string to screen. Do not use a nested loop.

If you did not do Q1 correctly, make this variable
`D1={'s1': ['g', 'a', 't'], 's2': ['t', 'a', 'c'], 's3': ['a']}`
and use D1 to do Q2.

2. **(20)** A blizzard is a massive snowstorm. Definitions vary, but for our purposes we will assume that a blizzard is characterized by both conditions:

winds of 30 mph or higher and blowing snow that leads to **visibility of 0.5 miles or less,**

sustained for at least four hours.

In a script called **ex2-blizzard.py** do the following:

Create a simulated data file of 4 lines of blizzard information for wind speed and visibility by **using a while loop**.
Within the while loop use:

- one input function to ask the user to enter a value for the wind of 30 mph or higher. The value is a real number.
- another input function to ask the user to enter a value for the visibility of 0.5 miles or less. Assume the value entered by the user is positive, and that is a real number.
- make an if statement to check the user's inputs meet the blizzard conditions
- write into a file called **blizzard.txt** the blizzard values both formatted to 1 decimal precision. Only the values that meet the blizzard conditions should be written into the file.

The data set should contain 4 correct blizzard values for the wind and visibility. To keep track of the number of correct values, you can incorporate a counting loop, or use other ways to count the correct values.
Open the file blizzard.txt in writing mode, and not appending mode.

As an example, the file blizzard.txt should look like this:

```
35.7 0.1
40.6 0.3
40.7 0.2
50.8 0.4
```

If you do not know how to write in the file, just print the 4 correct values to screen with 1 decimal precisions.

3. **(60)** In the provided script **ex3-datasets.py** there is a variable called **names**

```
names=['emissions.txt','labor.txt','land.txt','supercomputer.txt']
```

This variable contains the names of 4 different time series data sets. For each data set the values were collected during different years.

All these data sets are organized as follow:

- The first line is a header line, which reports information on the recorded topic, and its unit in this form:
  recorded topic - unit

  For example, for the supercomputer.txt the header is:
  Supercomputer power - FLOPS

- 1st field is the year
- 2nd field is a recorded value (a real number)
- The field separator is a comma.

The goal is to find, from the data, the maximum value and year in which the maximum value was recorded, and to extract the topic and its unit from the headerline. The code should work for all these data sets.

Edit the provided script and do the following:

Q1. (8) Make a function called ***find_max***. This function takes two parameters: a list of years and a list of the corresponding recorded numeric values. The function calculates the maximum recorded value and the year the maximum recorded value occurred, and returns a tuple containing both values. Use the max function, and index method.

Q2. (8) Make a function called ***topic_unit***. The function takes one headerline, which is one string of this form:

```
"recorded topic - unit"
```

The function uses a string method (for type conversion) and indexing to extract and then return a tuple of two strings *recorded topic* and *unit.* Think about how to separate the two strings and extract each of them.

If you apply the function to this headerline: "*Supercomputer power – FLOPS"*
the function should return a tuple of these two strings ("*Supercomputer power", "FLOPS")*

Q3. (6) Randomly pick one item from list **names** and assign that item to a variable called **filename**. Use a function from the random module.

If you do not know how to do this part make this variable

```
filename='supercomputer.txt'
```

Q4. (2) Use variable **filename** and print a sentence reporting the file you are analyzing.
In the print function, do not format, and do not concatenate. If you format or concatenate you will loss 2 points.
For example, in the case **supercomputer.txt** is selected, this sentence should be printed:

```
Analyzing supercomputer.txt
```

Q5. (3) Read in data from the data file stored in variable **filename**, and assign data to list L.
Use variable **filename**.

Q6. (5) Access values in list L, and store the headerline in variable **header**.
Access values in list L and store the data (from 2nd line to the end) in variable **L1**.
Do not hardcode the total number of lines.

Q7. (5) Use list comprehension to create a list of years **Ly** (1st field) out of list L1. This should be a list of integer numbers.

Q8. (5) Use list comprehension to create a list of values **Lv** (2nd field) out of list L1. This should be a list of float numbers.

Q9. (4) Call function ***find_max*** on lists **Lv** and **Ly** and store the returned values in two variables, M and Y. You should use tuple unpacking.

Q10. (4) Call function ***topic_unit*** on variable **header** and store the returned values in two variables **topi**c and **unit**. You should use tuple unpacking.

Q11. (10) Use the print function and variables **topic, M, unit** and **Y** and include strings to print to screen, for example:

```
Supercomputer power  max value 1.48e+17  FLOPS in 2019
```

Or other examples:

```
Labor Force Statistics  max value 9.7  percent in 1982

Global Land Temperature Anomalies max value 1.53 Celsius in 2016
```

You should remove the "\n" character which is at the end of the string stored in variable **unit**. For this, you can use a string method or slicing, within the print function.

If you concatenate or format within the print function, you will loss 3 points.

Extra credit (3) – all or none – hardcoding is not allowed.

Name the script **EC-s23.py**.
The file gardener.dat contains a poem from 'The Gardener' by Rabindranath Tagore. Your goal is to find all unique words that are 4 characters long and print those words to screen in alphabetic order. You should watch out for letter case, i.e., don't print to screen the same word with different letter case (uppercase, lowercase). Also make sure to only look at the words, i.e., to get rid of the following characters: **.,"!**

If you use python3 concepts or modules we have not discussed in class, you will not get credit for this problem.

```
bath
book
come
deep
eyes
felt
from
heat
hide
knee
long
look
near
noon
shut
when
with
```

Submission on Gradescope Exam2 the following:
ex1-s23.py
ex2-blizzard.py
ex3-datasets.py
EC-s23.py (if you made it)
DO NOT SUBMIT A ZIP FILE.