CMSC 201 Section 60
Project 1 - Attendance Records

Due Date: Monday, November 7th, 2022 by 11:59:59 PM
Value: 0 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student. Also, you should not post these problems on any forum, internet solutions website, etc.

```
"""
File:     FILENAME.py
Author:   YOUR NAME
Date:     THE DATE
Section:  YOUR DISCUSSION SECTION NUMBER
E-mail:   YOUR_EMAIL@umbc.edu
Description:
  DESCRIPTION OF WHAT THE PROGRAM DOES
"""
```

Project General Description

I wish I could tell you that every program you create will be a game. But most likely, it won't. But each of your instructors have been working on a fascinating project developing an application that did not include a game.

In this project, you will be given a lot of data that you will need to parse (look through). The data is from a new card swipe system in each classroom that tracks attendance. (Attending each class is important!) There are two pieces of data given:

1. Course roster for **one instance of a class**
    a. A simple list of names
2. Attendance records for **one instance of a class**
    a. A list of students that swiped using the system
    b. Contains:
        i. Student's name
        ii. Timestamp of their swipe **using military time**
        iii. Date of their swipe

You will also be introduced to something new in Python. You will be receiving two files. One file, dataEntry.py contains code that will fill the respective data arrays of the class roster and swipe data. You are welcome to use and change that file except for the function names. When we grade, we will replace that file with the same functions, but different values for both the roster and swipe data. The second file, p1.py, has a lot of code already in it. **There are portions of the file that WILL NOT be altered in ANY manner.** One portion of the file includes the dataEntry.py swipe data file. The other part is the main.

Your focus is to finish all of the functions listed on p1.py. These functions manipulate the data to get query data. Each “stub” has comments in order to help you develop the solution. You are expected to follow the design of the function commented with the function header. The function should be able to be called by the unchanged function calls within the main(). We also show the expected output below.

Provided Files

I have provided a starter file on the GL server at:

/afs/umbc.edu/users/s/l/slupoli/pub/cmsc201/fall/dataEntry.py
/afs/umbc.edu/users/s/l/slupoli/pub/cmsc201/fall/p1.py

You should copy it to your directory using the command:

cp /afs/umbc.edu/users/s/l/slupoli/pub/cmsc201/fall22/dataEntry.py dataEntry.py
cp /afs/umbc.edu/users/s/l/slupoli/pub/cmsc201/fall22/p1.py p1.py

Submission Details

Submit the files under the following titles:
(These are case sensitive as usual.)
submit cmsc201 PROJECT1 p1.py

You only need to submit your p1.py file since we will use our own dataEntry.py.

Coding Standards

Coding standards can be found here.

1. At least one inline comment per function explaining something about your code.
2. Constants above your function definitions, outside of the "if __name__ == '__main__':" block.
    a. A magic value is a string which is outside of a print or input statement, but is used to check a variable, so for instance:
        i. `print(first_creature_name, 'has died in the fight. ')` does not involve magic values.
        ii. However, `if my_string == 'EXIT':` exit is a magic value since it's being used to compare against variables within your code, so it should be:
        ```
        EXIT_STRING = 'EXIT'
        if my_string == EXIT_STRING:
        ```
    b. A number is a magic value when it is not 0, 1, and if it is not 2 being used to test parity (even/odd).
    c. A number is magic if it is a position in an array, like my_array[23], where we know that at the 23rd position, there is some special data. Instead it should be

```
USERNAME_INDEX = 23
my_array[USERNAME_INDEX]
```

   d. Constants in mathematical formulas can either be made into official constants or kept in a formula.
3. Previously checked coding standards involving:
   a. snake_case_variable_names
   b. CAPITAL_SNAKE_CASE_CONSTANT_NAMES
   c. Use of whitespace (2 before and after a function, 1 for readability.)

Allowed Built-ins/Methods/etc

- Declaring and assigning variables, ints, floats, bools, strings, lists.
- Using +, -, *, /, //, %, **; +=, -=, *=, /=, //=, %=, **= where appropriate
- Comparisons ==, <=, >=, >, <, !=, in
- Logical and, or, not
- if/elif/else, nested if statements
- Casting int(x), str(x), float(x), (technically bool(x))
- For loops, both *for i* and *for each* type.
- While loops
  - sentinel values, boolean flags to terminate while loops
- Lists, list(), indexing, i.e. my_list[i] or my_list[3]
  - 2d-lists if you want them/need them my_2d[i][j]
  - Append, remove, del
  - **list slicing**
- If you have read this section, then you know the secret word is: createous.
- String operations, concatenation +, +=, split(), strip(), join(), upper(), lower(), isupper(), islower()
  - **string slicing**

- Print, with string formatting, with end= or sep=:
    - '{}'.format(var), '%d' % some_int, f-strings
    - Really the point is that we don't care how you format strings in Python
    - Ord, chr, but you won't need them this time.
- Input, again with string formatting in the prompt, casting the returned value.
- Using the functions provided to you in the starter code.
- Using import with libraries and specific functions **as allowed** by the project/homework.
- You may define your own new functions with new parameters
    - Single return values (you must return a single float, bool, string, int, list or dictionary, or None/not return).
- **For this project, the sum keyword is permitted.**
- **For this project, dictionaries are permitted but not necessary.**

Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

- Multiple returns [i.e. `var1, var2 = function_call()`]
- break, continue
- methods outside those permitted within allowed types
    - for instance str.endswith
    - list.index, list.count, etc.
- Keywords you definitely don't need: await, as, assert, async, class, except, finally, global, lambda, nonlocal, raise, try, yield
- The *is* keyword is forbidden, not because it's necessarily bad, but because it doesn't behave as you might expect (it's not the same as ==).
- built in functions: any, all, breakpoint, callable, classmethod, compile, exec, delattr, divmod, enumerate, filter, map, max, min, isinstance, issubclass, iter, locals, oct, next, memoryview, property, repr, reversed, set, setattr, sorted, staticmethod, super, type, vars, zip.
- If you have read this section, then you know the secret word is: serendipity.
- The two built in functions exit() and quit()
- If something is not on the allowed list, not on this list, then it is probably forbidden.
- The forbidden list can always be overridden by a particular problem, so if a problem allows something on this list, then it is allowed for that problem.
- Note that we are using python 3.9.7 currently on the GL server so new features like switch/case are 'forbidden' but only in the sense that it doesn't exist in our current version.

Sample Project Output by Function

For each function, there is a partial function header. You are required to fill the param and return details as a part of your grade. Use the given code in the main and the function descriptions to help you determine those details. The description will be given, but what parameters will the function accept will have hints from the main() given.

def list_students_not_in_class(???):

Compare the swipe log with the given course roster. Place those students that did not show up for class into a list.

```
linux3[25]% python3 p1.py

...
****** Students missing in class *************
Arsenault, Al
Mammel, Sammie
Sheldon, Simon
...
```

def list_all_times_checking_in_and_out(???):

Looking at the swiping log, this function will list all in and outs for a particular Student. Please note, as coded in the p1.py file given, this function was called three times with different student values. The last example (called) used a student that skipped class. (Al Arsenault… shame!)

```
linux3[25]% python3 p1.py

...
****** List all swipe in and out for a student *******
Lupoli, Shawn, 08:55:14, 10/20/2022
Lupoli, Shawn, 09:20:14, 10/20/2022
****** List all swipe in and out for a student *******
Allgood, Nick, 09:02:14, 10/20/2022
****** List all swipe in and out for a student *******
********************************

...
```

def list_all_times_checked_in(???):

This function returns a list of when all student(s) FIRST swipe in.

```
linux3[25]% python3 p1.py

...
****** Check in times for all students who attended***
Chen, Kevin, 08:55:34, 10/20/2022
Diaz, Sergio, 08:55:44, 10/20/2022
Dulce, Alexander John, 08:55:54, 10/20/2022
Gaither, Oliver, 08:56:10, 10/20/2022
Hambrecht, Matt, 08:56:11, 10/20/2022
Hassanein, Magdi, 08:56:12, 10/20/2022
Henderson, Sean, 08:56:13, 10/20/2022
Lare, George, 08:56:14, 10/20/2022
Hamilton, Eric, 08:50:34, 10/20/2022
Lupoli, Shawn, 08:55:14, 10/20/2022
Aja, Richard, 08:55:14, 10/20/2022
Barot, Bharg, 08:55:24, 10/20/2022
Odeseye, Ayodele, 08:57:14, 10/20/2022
Ortiz, Joseph, 08:57:15, 10/20/2022
Othman, Youssef, 08:57:16, 10/20/2022
...
Asad, Hamza, 09:03:14, 10/20/2022
Cajudoy, Janeiyah, 09:04:14, 10/20/2022
Cleary, Miranda, 09:05:14, 10/20/2022
Dove, Ellie, 09:06:14, 10/20/2022
Gunaseelan, Adith, 09:07:14, 10/20/2022
Ilamni, Jazlyn, 09:02:15, 10/20/2022
Jacob, Ify, 09:02:16, 10/20/2022
Pham, Matthew, 09:02:17, 10/20/2022
Tang, Hieu, 09:02:18, 10/20/2022
Vantran, Luke, 09:02:21, 10/20/2022
Vedasendur Senthilvel, Pranav, 09:02:22, 10/20/2022
Zhuang, Michael, 09:02:23, 10/20/2022
```

def list_students_late_to_class(???):

When given a timestamp string and the swipe log, a list of those records of students who swiped in late into the class is produced. Do not record students who swiped out after the inputted timestamp.

```
linux3[25]% python3 p1.py

...
****** Students that arrived late ********************
Allgood, Nick, 09:02:14, 10/20/2022
Asad, Hamza, 09:03:14, 10/20/2022
Cajudoy, Janeiyah, 09:04:14, 10/20/2022
Cleary, Miranda, 09:05:14, 10/20/2022
Dove, Ellie, 09:06:14, 10/20/2022
Gunaseelan, Adith, 09:07:14, 10/20/2022
Ilamni, Jazlyn, 09:02:15, 10/20/2022
Jacob, Ify, 09:02:16, 10/20/2022
Pham, Matthew, 09:02:17, 10/20/2022
Tang, Hieu, 09:02:18, 10/20/2022
Vantran, Luke, 09:02:21, 10/20/2022
Vedasendur Senthilvel, Pranav, 09:02:22, 10/20/2022
Zhuang, Michael, 09:02:23, 10/20/2022
...
```

def get_first_student_to_enter(???):

Simply, this function returns the student that swiped in first. Note, the order of the data entered into the swipe log as not the earliest student to enter.

```
linux3[25]% python3 p1.py

...
******* Get 1st student to enter class ****************
Hamilton, Eric
...
```

def printList(???):

This function simply prints the list. The function should not be able to change any values of that list passed in. You will notice the main() uses this function often.