Project 3 - RetrieverBell

Due Date: Friday, Wednesday December 13th, 2023
by 11:59:59 PM
Value: 90 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.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
"""
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
"""
```

Submission Details

Submit the files under the following title:
(These are case sensitive as usual.)

When you're finished with the project submit as:
submit cmsc201 PROJECT3 network.py

Project Description

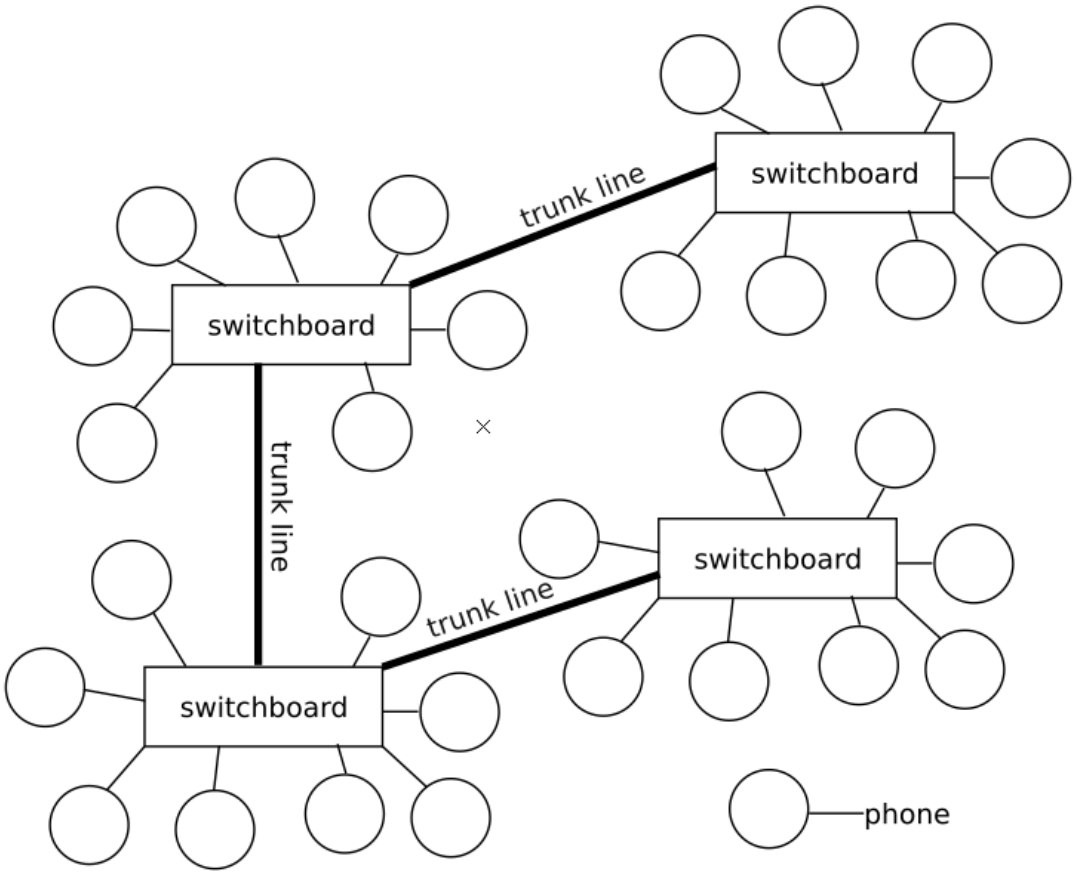
Because of a new surprise round of antitrust cases, a new phone company RetrieverBell was created. You've been hired as the single programmer whose duty is to code the connection servers for all of the phones signed up with your company.

Like the old days before cell phones, phones will have an area code, which is permitted to be any positive number of digits and a phone number, which is permitted to be any positive number of digits.

In order to connect a phone to another phone, if it's a local number in the same area code, it can connect through its local switchboard. On the other hand, if it's a long distance number, it has to connect through the connected switchboards.

Here's an old switchboard system.

A Sample Diagram of Our Network



Trunk lines allow us to connect between switchboards and each phone is connected to exactly one switchboard (and has exactly one area code).

In our network, there is a 1-to-1 correspondence between switchboards and area codes. (I.e. each switchboard has one area code attached, and that area code is fully handled by that board.)

Implementation Details

The required functionality for the project won't be described exactly in the way you have to implement it, but I will specify operations and what they need to do.

switch-add

switch-add [area-code-1]

This will create a new switchboard with the area code specified. The braces are just to indicate that it needs to be replaced with an area code which is a number. If the switchboard exists, you don't create a new one.

switch-connect

switch-connect [area-code-1] [area-code-2]

This connects the switchboard at [area-code-1] with the one at [area-code-2], and also the reverse connection, which means that they are connected to each other. Calls can flow in both directions.

phone-add

phone-add [area-code-1]-[num_part_1]-[num_part_2]

phone-add should add a phone number with the area code specified as well as the phone number with one hyphen between the area code and the phone number, but there can be as many hyphens between any other part of the number. The rest of the number can be combined, or kept as a format with hyphens.

start-call

start-call [phone_number_1] [phone_number_2]

Connecting a call requires a recursive function. It can be a helper function rather than start_call()

start-call should create a connection between the two phone numbers, if the area codes exist, the numbers exist at their proper area codes, and if they are not already in a phone call. It can only be connected if there is a path along switchboards between the two numbers. It is possible that no connection will exist because the two switchboards could not be connected by "trunk lines."

end-call

end-call [phone_number]

end-call should disconnect the two numbers that are connected. The disconnection can occur at either end of the call. If there's no connection from that number, then report that but you don't need to do anything else.

display

display

The display command should output each switchboard with its area-code. Then output the other area codes/switchboards that it's connected to. Finally output each local phone number and if it is on the hook, or if it's connected to another phone. See the sample output for more on how I think it should generally look to make it readable for the console output.

network-save

network-save [filename]

This should save the network in a file. You can choose the format that you use, and you are permitted to use csv, json packages if you wish. You must decide on the file format, which must be able to be saved, and then loaded in a different run of the program. The structure of the network should be saved, but the phone to phone connections can be lost (pretend that all phones are hung up at the start of the load).

network-load

network-load [filename]

This should load the file at file-name. It should create the network described in the file. Remember that you have to specify the file format, so you'll need to decide how to save, and how to load. **When you load a network, you should delete the old network or overwrite it, whichever makes sense for your project.**

You will probably have to modify the main driver code to check for safety conditions and display output messages.

Starter Code

One file is provided on the GL server for this:

/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall23/network_starter.py

When you copy this file to your computer, you must rename it network.py.

You will probably have to modify the main driver code to check for safety conditions and display output messages. You are also permitted, encouraged, and probably required to add additional helper functions.

Remember that in order to copy a file you cannot copy and paste the path by itself. You must do cp <the path> . The period indicates that you're copying to the current directory.

Sample Output 1

I've provided two sample output runs. They are far from complete, meaning they don't cover nearly all possible things that can happen. Your output especially in display doesn't have to match exactly but it should be readably similar.

```
linux3[1729]% python3 network.py
Enter command: switch-add 443
Enter command: switch-add 410
Enter command: switch-add 656
Enter command: display
Switchboard with area code:  443
    Trunk lines are:
    Local phone numbers are:
Switchboard with area code:  410
    Trunk lines are:
    Local phone numbers are:
Switchboard with area code:  656
    Trunk lines are:
    Local phone numbers are:
Enter command: switch-connect 443 410
Enter command: phone-add 656-112-3412
Enter command: phone-add 443-132-1332
Enter command: display
Switchboard with area code:  443
    Trunk lines are:
       Trunk line connection to: 410
    Local phone numbers are:
       Phone with number: 1321332 is not in use.
Switchboard with area code:  410
    Trunk lines are:
       Trunk line connection to: 443
    Local phone numbers are:
Switchboard with area code:  656
    Trunk lines are:
    Local phone numbers are:
       Phone with number: 1123412 is not in use.
Enter command: start-call 656-112-3412 443-132-1332
656-112-3412 and 443-132-1332 were not connected.
Enter command: switch-connect 656 410
Enter command: start-call 656-112-3412 443-132-1332
656-112-3412 and 443-132-1332 are now connected.
Enter command: display
Switchboard with area code:  443
```

```
        Trunk lines are:
           Trunk line connection to: 410
        Local phone numbers are:
           Phone with number: 1321332 is connected to 656-1123412
    Switchboard with area code:  410
        Trunk lines are:
           Trunk line connection to: 443
           Trunk line connection to: 656
        Local phone numbers are:
    Switchboard with area code:  656
        Trunk lines are:
           Trunk line connection to: 410
        Local phone numbers are:
           Phone with number: 1123412 is connected to 443-1321332
    Enter command: network-save sample1.net
    Network saved to sample1.net.
    Enter command: quit
```

Sample Output 2

```
linux3[1730]% python3 network.py
Enter command: network-load test.net
Network loaded from test.net.
Enter command: display
Switchboard with area code:  310
    Trunk lines are:
       Trunk line connection to: 410
    Local phone numbers are:
       Phone with number: 2222222 is not in use.
       Phone with number: 5555555 is not in use.
       Phone with number: 1111111 is not in use.
       Phone with number: 1231233 is not in use.
Switchboard with area code:  410
    Trunk lines are:
       Trunk line connection to: 310
       Trunk line connection to: 443
    Local phone numbers are:
       Phone with number: 2233333 is not in use.
       Phone with number: 3453456 is not in use.
Switchboard with area code:  443
    Trunk lines are:
       Trunk line connection to: 410
    Local phone numbers are:
       Phone with number: 9109191 is not in use.
Switchboard with area code:  545
    Trunk lines are:
    Local phone numbers are:
       Phone with number: 1231234 is not in use.
Switchboard with area code:  223
    Trunk lines are:
    Local phone numbers are:
       Phone with number: 1231234 is not in use.
Enter command: start-call 223-1231234 410-223-3333
223-1231234 and 410-223-3333 were not connected.
Enter command: start-call 410-223-3333 310-555-5555
410-223-3333 and 310-555-5555 are now connected.
Enter command: start-call 310-222-2222 310-111-1111
310-222-2222 and 310-111-1111 are now connected.
Enter command: display
Switchboard with area code:  310
    Trunk lines are:
       Trunk line connection to: 410
    Local phone numbers are:
```

```
        Phone with number: 2222222 is connected to 310-1111111
        Phone with number: 5555555 is connected to 410-2233333
        Phone with number: 1111111 is connected to 310-2222222
        Phone with number: 1231233 is not in use.
Switchboard with area code:  410
    Trunk lines are:
        Trunk line connection to: 310
        Trunk line connection to: 443
    Local phone numbers are:
        Phone with number: 2233333 is connected to 310-5555555
        Phone with number: 3453456 is not in use.
Switchboard with area code:  443
    Trunk lines are:
        Trunk line connection to: 410
    Local phone numbers are:
        Phone with number: 9109191 is not in use.
Switchboard with area code:  545
    Trunk lines are:
    Local phone numbers are:
        Phone with number: 1231234 is not in use.
Switchboard with area code:  223
    Trunk lines are:
    Local phone numbers are:
        Phone with number: 1231234 is not in use.
Enter command: end-call 310-111-1111
Hanging up...
Connection Terminated.
Enter command: end-call 545-123-1234
Unable to disconnect.
Enter command: end-call 410-555-5555
Enter command: display
Switchboard with area code:  310
    Trunk lines are:
        Trunk line connection to: 410
    Local phone numbers are:
        Phone with number: 2222222 is not in use.
        Phone with number: 5555555 is connected to 410-2233333
        Phone with number: 1111111 is not in use.
        Phone with number: 1231233 is not in use.
Switchboard with area code:  410
    Trunk lines are:
        Trunk line connection to: 310
        Trunk line connection to: 443
    Local phone numbers are:
        Phone with number: 2233333 is connected to 310-5555555
        Phone with number: 3453456 is not in use.
Switchboard with area code:  443
```

```
        Trunk lines are:
           Trunk line connection to: 410
        Local phone numbers are:
           Phone with number: 9109191 is not in use.
    Switchboard with area code:  545
        Trunk lines are:
        Local phone numbers are:
           Phone with number: 1231234 is not in use.
    Switchboard with area code:  223
        Trunk lines are:
        Local phone numbers are:
           Phone with number: 1231234 is not in use.
    Enter command: quit
```

Generalized Rubric

I've listed the approximate point values for each part of functionality in the project. They are subject to change slightly but this will generally be the rubric that we adhere to. We will break each one of these down into sub-parts which will have point values which add up to close to the approximate point value.

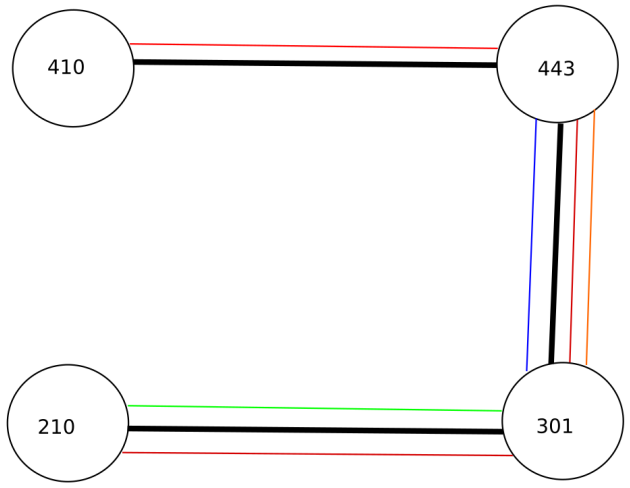
Functionality	**Apprx. Point Value**
switch-add functionality	5
display functionality	5
switch-connect functionality	10
phone-add functionality	10
start-call functionality	15
end-call functionality	15
network-save/load functionality	20
Coding Standards	10
Limited lines extra credit	10

Extra Credit: Limited Phone Lines

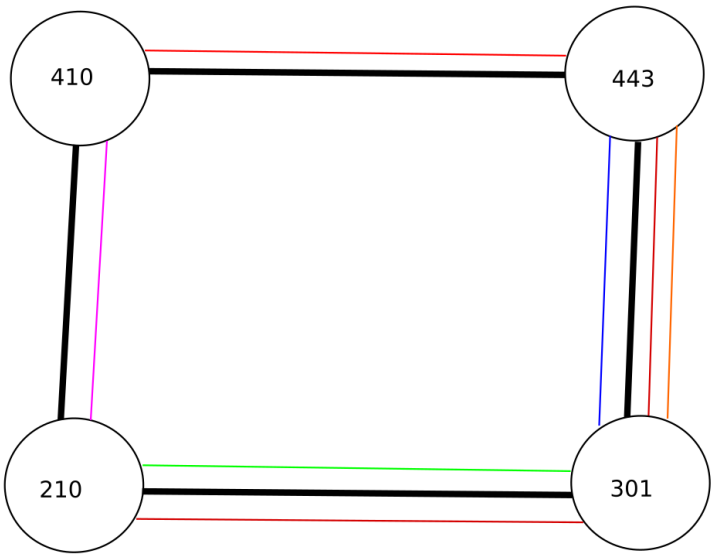
For the extra credit you must implement a bit of additional functionality. We are going to ask before the program starts for a maximum number of phone calls that can be put on a single trunk line at any time. Then your job will be to only allow that number of calls to exist maximally on any given line.

This means that every line along the path must be incremented to ensure that you don't exceed the capacity of any given line. Then, you can return that line to idle if the call hangs up. This will require tracing along the path of the call and freeing up a used line along each of the trunk connections.

Consider the diagram here, imagining that the limit of trunk connections is 3. The thick black lines are trunks, the red line is a phone call from 410 to 210 area codes. Blue is a call from 443 to 310, as is orange. Green is a call from 301 to 201. Now, note that no further calls can be entered from 410 to 210 or back because the lines are all used up in between.



On the other hand, if we introduce another path:



We can introduce another call, the magenta line from 410 to 210 because there are open lines now.

My hint for this would be once you're finished with the project, you may have to add a bit more structure to the neighbors list or have a duplicate neighbors list that contains the number of connections currently used.

Coding Standard: Global Variables

Global variables **are not** permitted for this project beyond a creation of variables in the if __name__ == '__main__': but those variables should be passed into functions or class methods. They should not be accessed from the global scope.

A global variable is anything declared outside of a class, outside of a method, i.e. not in a class scope or method scope.

Global constants **are permitted** for this project.

This means you can still use global constants at the top of your program, which is recommended.

Coding Standards

Coding standards can be found here.

We will be looking for:

1. At least one inline comment per program 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, dicts.
- 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
    - **list slicing**
- If you have read this section, then you know the secret word is: enfilade.
- 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.

- **Dictionaries**
    - creation using dict(), or {}, copying using dict(other_dict)

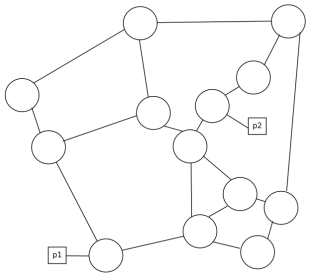
    - .get(value, not_found_value) method
    - accessing, inserting elements, removing elements.
- Using the functions provided to you in the starter code.
- Using import with libraries and specific functions **as allowed** by the project/homework.
- **Recursion - allowed and required for this project.**
- Tuples are allowed, as they are immutable lists.
- json and csv libraries are allowed and optional for project 3.
- File read/write methods:
    - f.read(), f.readlines(), f.readline()
    - f.write(), f.writelines()
    - open(filename, 'r' or 'a', or 'w')
    - with open(filename, mode) as the_file:
    - f.close() if you don't use with.

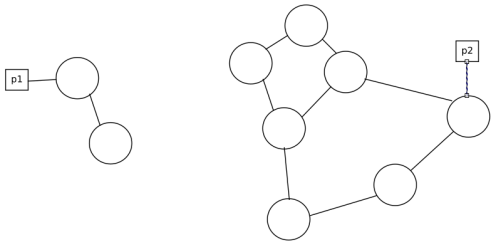
Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

- 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, assert, async, 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, next, memoryview, property, repr, reversed, round, set, setattr, sorted, staticmethod, sum, super, type, vars, zip
- If you have read this section, then you know the secret word is: quizzical.
- exit() or 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.

Frequently Asked Questions

- Can I modify X method? Change the parameters? Add a class? Redesign the basic project design? Modify the driver code?
    - Yes.
- What kind of file type do we need?
    - You are designing the file type.
- Are there sample network files?
    - No, see above question.
- Why do we need recursion for the project?
    - Because you need to be able to tell the difference between these two situations:
    - 

    - 

    - In the first case, the phones can connect, but in the second case they can't. Telling these two apart is not as easy as it would seem if you try a purely iterative solution. In general it isn't possible to do without a recursive solution or more complex algorithms.