Zombie Contact Tracing

Overview

In this assignment you will analyze fictitious contact tracing data. This data is one-directional, meaning that people who have tested positive for a fictitious zombie disease will report the people with which they have had contact. There will not be data for the reverse direction (i.e., where a person had contact with a sick individual).

From the data, you will report various types of higher-level information (these will be detailed later):

- Contact records;
- Patient zeros;
- Potential zombies;
- Neither patient zeros nor potential zombies;
- Most viral people;
- Most contacted people; and
- Maximum distance from potential zombies.

There will also be a few additional categories that can earn you extra credit if you address them.

Input file format

In order to work with the contract tracing data, you will need to obtain input from a data file containing contact tracing data. You can obtain this data from *stdin* (which is the preferred method) or by opening a file in your program.

The data is stored in CSV (comma separated value) files. Each line in the file is formatted as follows:

```
SickPerson,Contact1,Contact2,...,ContactN
```

Where `SickPerson` and `ContactX` are placeholders for specific people.

Each line will always begin with exactly one person, followed by one or more other people. Here are some examples of lines in one data file:

```
Bob,Paul,Mark,Carol,Leanne,Will
```

```
Carol,Mark,Leanne
```

Note that names may include a mixture of upper and lowercase letters, and may also include spaces. In order to make the data files easier to work with, you can assume that there will not be any spaces immediately before or after any of the commas.

Specification

Write a Python program that reads contact tracing data from an input file, analyzes it, and produces output that summarizes various characteristics about the data. You may implement each characteristic in its own function (or not – it's up to you – as long as it makes sense and exemplifies good coding style). Whichever you choose, make sure to clearly comment each part.

The question of which data structure(s) to use to store the data in order to efficiently produce the desired output is also up to you. Feel free to use any of Python's internal data structures (e.g., list, dictionary, etc) or those that we covered (and probably coded) in class (e.g., unordered list, stack, priority queue, etc). **Make sure to justify why you chose specific data structure(s) in your comments.** Specifically, note why you used one or more data structure(s) to address each part identified below. **You will be graded on this.**

Part 1: Who did each sick person have contact with?

Your first task is to list everyone that each sick person had contact with. For example, if the first two lines of data contain the following:

```
Bob,Paul,Mark,Carol,Leanne,Will
Carol,Mark,Leanne
```

Your output should be:

```
Bob had contact with Carol, Leanne, Mark, Paul, Will
Carol had contact with Leanne, Mark
```

Pay close attention to the sample output so that you structure yours exactly the same as above. Also note that the contact list is sorted by name!

For reference, the output for this part of the input file **DataSet1.txt** should be:

```
Contact records:
  Bob had contact with Carol, Leanne, Mark, Paul, Will
  Carol had contact with Leanne, Mark
  Farley had contact with Paul
  Larry had contact with Carol, Leanne, Mark, Will
```

```
Leanne had contact with Sarai
Mark had contact with Philip, Zach
Paul had contact with Zach
Will had contact with Leanne, Mark
Zach had contact with Philip
```

Again, note the output (including the header “Contact records:” and indentation (two spaces) before each line representing who a sick person had contact with).

Part 2: Who are the patient zeros?

One way to think of patient zeros is that they are people who are sick but who do **not** appear in anyone else's contact list. Therefore, they only appear as the initial name in a line of input!

Your program should display the patient zeros as follows (for **DataSet1.txt**):

```
Patient zeros: Bob, Farley, Larry
```

Part 3: Who are potential zombies?

We will define a potential zombie to be any person that **might** be infected (because they have been in contact with a sick individual) but who has not yet been conclusively determined to be sick. Remember that a sick person occurs as the first name in each contact record in the data file; therefore, a potential zombie will be in the contact list of a sick person, but never be the initial name in a line of input.

Your program should display the potential zombies as follows (for **DataSet1.txt**):

```
Potential zombies: Philip, Sarai
```

Part 4: Who are neither patient zeros nor potential zombies?

Obviously, this is everyone who is not listed in parts 2 or 3.

Your program should display these individuals as follows (for **DataSet1.txt**):

```
Neither patient zero nor potential zombie: Carol, Leanne, Mark,
Paul, Will, Zach
```

Part 5: Who are the most viral people?

We will define the most viral people as the people who likely infected the greatest number of other people in the data set because they had the longest list of contacts.

Your program should display these individuals (there may be more than one) as follows (for **DataSet1.txt**):

```
Most viral people: Bob
```

Part 6: Who are the most contacted people?

We will define the most contacted people as those who have been infected by the most sick people. Obviously, these are the people who appear in the most contact records.

Your program should display these individuals as follows (for **DataSet1.txt**):

```
Most contacted people: Leanne, Mark
```

Part 7: What is each person's maximum distance from a potential zombie?

We will define the maximum distance from a potential zombie as the longest contact tracing path downwards in the data set from a sick person to a potential zombie. Using this definition, all potential zombies (people that are not yet confirmed to be sick) have a maximum distance of 0. Any person who only has contact with potential zombies has maximum distance of 1. All other people in the data set have a maximum distance which is one more than the maximum distance of the person that they have had contact with – who has the largest maximum distance value.

One way to think about this is that the description of maximum distance is **recursive**. It defines the maximum distance of one person in terms of the maximum distance of another person. As such, you will probably want to write a recursive function to determine the maximum distance of each person in the contact tracing data.

Your program should display the maximum distances as follows (for **DataSet1.txt**):

```
Maximum distance from a potential zombie:
  Bob: 4
  Larry: 4
  Carol: 3
  Farley: 3
  Will: 3
  Mark: 2
  Paul: 2
  Leanne: 1
  Zach: 1
  Philip: 0
  Sarai: 0
```

Extra credit:

For extra credit, you must identify the following additional categories:

- **Spreader zombies**: a spreader zombie is a sick person that has only had contact with potential zombies (i.e., people who haven't yet been determined to be sick – and aren't the initial name in a line of input).
- **Regular zombies**: a regular zombie is a sick person that has had contact with both potential zombies and people who are already sick.
- **Zombie predators**: a zombie predator is a person that has only had contact with people who are sick (i.e., people who have been determined to be sick – and appear as the initial name in a line of input).

For **DataSet1.txt**, the output should be as follows:

```
For extra credit:
  Spreader zombies: Leanne, Zach
  Regular zombies: Mark
  Zombie predators: Bob, Carol, Farley, Larry, Paul, Will
```

Constraints and notes

Note the following constraints and notes (see the rubric below for more detail):

- Your program must either obtain input from *stdin* (which is preferred) or by opening a data file (e.g., **DataSet0.txt**). If you choose the latter, **do not hard code paths** (i.e., just specify the input file in the current directory).
- Submit a **single** program that addresses all parts (and the extra credit if applicable).
- You must choose the appropriate data structure(s) for each part. It is possible that a single data structure can be used for multiple parts. In any case, make sure to add comments in your code that justifies the data structure(s) that you use for each required part. If you attempt extra credit, justify them there too.
- Feel free to use built-in Python data structures or any that we designed in class.
- You must compute the maximum distances from potential zombies recursively.
- Use functions appropriately (but don't force them when/where they don't need to be).
- You must use good coding style (which includes things such as including an informative header at the top of your program, commenting your source code appropriately, using meaningful variable and (if applicable) constant identifiers, and so on).

Deliverable

Submit a Python 3 source file that can be executed through an IDE (such as Thonny) or via the command line (e.g., `python3 Zombies.py`).

Sample output

To make sure that your implementation of each part is correct, sample output will be provided in a text file. However, here is (complete) sample output for **DataSet5.txt**:

```
Contact records:
  Gerald had contact with Paolo
  Jack had contact with Gerald, Robert
  Karl had contact with Mario, Shay
  Lonnie had contact with Gerald, Jack, William III
  Mario had contact with Paolo
  Olly had contact with Mario
  Robert had contact with Paolo
  Shay had contact with Mario
  William III had contact with Mario, Robert

Patient zeros: Karl, Lonnie, Olly
Potential zombies: Paolo
Neither patient zero nor potential zombie: Gerald, Jack, Mario,
Robert, Shay, William III
Most viral people: Lonnie
Most contacted people: Mario

Maximum distance from a potential zombie:
  Karl: 3
  Lonnie: 3
  Jack: 2
  Olly: 2
  Shay: 2
  William III: 2
  Gerald: 1
  Mario: 1
  Robert: 1
  Paolo: 0
```

```
For extra credit:
  Spreader zombies: Gerald, Mario, Robert
  Regular zombies: None
  Zombie predators: Jack, Karl, Lonnie, Olly, Shay, William III
```

For extra credit, note the output `None` if there are no people in a category.

Hints

Here are some hints to help get you started:

- Of course, feel free to Google things and/or discuss high-level ideas with your classmates. As always, do not Google solutions and make sure to **cite sources** (including your classmates).
- Develop an algorithm for each part before trying to write the code.
- Structure your program so that it first reads the input data and stores it into one or more appropriate data structures. Then work on each of the required parts.
- Obviously, begin with part 1 since all of the other parts effectively rely on it. Don't move on to the other parts until you have the correct output for part 1.
- Parts 2 through 6 can be performed in any order. Part 4 is probably the easiest to complete. Part 6 is probably the most difficult.
- If you attempt the extra credit, don't do so until you have correctly addressed all of the required parts.

Rubric

Please note the following rubric for this programming assignment:

Zombie Contact Tracing / SciFiLi		
#	**Item**	**Points**
1	Good coding style	5
2	Input properly obtained	5
3	Data structures adequately justified	15
4	Recursive requirements addressed	5
5	Functions appropriately used	5
6	All parts correctly addressed	15
TOTAL		**50**