CSE 20
Beginning Programming in Python
Lab Assignment 5

In this project you will write yet another Python script to be run on the Unix timeshare, called `Search`. This program will emulate the most elementary functionality of the `grep` command in Unix. The command

```
$ grep <string> <file>
```

where `<string>` is a string and `<file>` is a text file in your current working directory, prints all lines in file `<file>` that contain the substring `<string>`. Depending on your terminal settings, it may also highlight the occurrences of `<string>` in a different color. Download the files `Tolkien1` and `Tolkien2` from `/Examples/lab5`, and place them in a convenient directory on the timeshare (like `~/cse20/lab5`). From within that directory, type

```
$ grep Ring Tolkien1
```

and observe that all lines containing instances of the string `'Ring'` are printed. Notice that there are 4 instances, and one instance is within the word "Rings". Thus, we see that `grep` is finding substrings of each line, considered as a large string. You can also search for strings that contain spaces, as long as you enclose that string in quotes (which you may do in any case.) Try doing

```
$ grep 'nd t' Tolkien1
```

to find two instances. Any string that contains characters having a special meaning to the bash interpreter, such as the space, must be enclosed in quotes. For instance, if you search for a semi-colon `;` by doing

```
$ grep ; Tolkien2
```

the interpreter thinks you are running two commands: `grep` (which gives you a usage message because you gave it no arguments), followed by `Tolkien2` (which gives a warning because the file is not executable). On the other hand

```
$ grep ';' Tolkien2
```

sucsessfully locates two occurrences of `;` within `Tolkien2`.

The command `grep` stands for: **G**lobally search for a **RE**gular expression and **P**rint matching lines. A *regular expression*, or *regex*, is a sequence of characters that specifies a search pattern. The simplest possible such pattern is just a string to be searched for and matched, as demonstrated in the above examples. More complex regular expressions can be used by `grep` to match *sets of strings* rather than just a single string. We will not cover more advanced regular expressions here. Your `Search` program will emulate the functionality of the `grep` on these elementary examples. Google the term *regular expression* to learn more about this important concept, or do `man grep` to learn more about the functionality of the `grep` command.

The output of `Search` will differ from that of `grep` in an essential way however. Instead of printing out the lines with matching occurrences of `<string>`, `Search` will state the line number and column number of (the beginning of) each occurrence of `<string>` within `<file>`. We illustrate the operation of `Search` below on all of the preceding examples, and a few more.

```
$ Search Ring Tolkien1

'Ring' found at:
   line 1, column 7
   line 6, column 5
   line 6, column 32
   line 7, column 5

$ Search 'nd t' Tolkien1

'nd t' found at:
   line 6, column 42
   line 7, column 50

$ Search ';' Tolkien2

';' found at:
   line 2, column 26
   line 8, column 5

$ Search Ring Tolkien2

'Ring' not found

$ Search Ring
Usage: Search <string> <file>
$ Search
Usage: Search <string> <file>
$
```

As always, your output must match the above format exactly for full credit. Observe that if the user does not have exactly two command line arguments after the Search command, a usage message will be printed.

The basic algorithm for this program will be to get each line of `<file>`, and search for all occurrences of the substring `<string>` within that line. When you find a match, save the line number and the position in the line at which the substring begins. When all lines have been searched, print out your match positions in the proper format, or print a message stating that the substring was not found. Note that when counting lines and columns, your counts must begin at 1, not at 0. See the following examples to learn different ways to iterate over the lines in a file.

```
/Examples/File1.py
/Examples/File2.py
/Examples/File3.py
/Examples/File4.py
/Examples/pa6/FileCopy.py
```

Also, see the documentation of function `str.find()` at

https://docs.python.org/3.9/library/stdtypes.html#text-sequence-type-str

to learn how to search a single line for a substring. Follow the general template posted in `/Examples` when writing this program. As always start early and ask for help if anything is not fully clear. Submit your file `Search` to lab5 on Gradescope before the due date.