# Computer Science 1 — CSci 1100
# Lab 7 — Easter Egg Hunt

## Lab Overview

This lab uses an Easter egg hunt to review the use of files. Please download the file `lab07_files.zip` from the Piazza site. This includes a set of large text files named `1.txt`, `2.txt`, etc. Inside these files are hidden Python programs, total 6 of them. To hunt them down, you need to recover the individual lines and write them down into a program file. Then, you must execute that file within Wing. You will have to do a bit of file parsing to get there. Collect them all!

## Checkpoint 1: More Text Processing Practice

Write a function called `parse_line` that takes as input a line of text such as

```
Here is some random text, like 5/4=3./12/3/4
```

and converts this into a four tuple of the form:

```
(12,3,4,"Here is some random text, like 5/4=3.")
```

where the last three values are converted to integers, and the remainder of the text is a string. The last values are always assumed to be separated by a slash.

You must however do some error checking. If you do not have a line that has at least three `'/'` in this given format or if the values contained in the last three slots are not integers, then your function should return `None`. Here are some example runs of this function:

```
>>>parse_line("Here is some random text, like 5/4=3./12/3/4")
(12, 3, 4, 'Here is some random text, like 5/4=3.')
>>>parse_line("Here is some random text, like 5/4=3./12/3/4as")
None
>>>parse_line("Here is some random text, like 5/4=3./12/412/a/3/4")
None
>>>parse_line("     Here is some spaces 12/32/4")
None
>>>parse_line("    Again some spaces\n/12/12/12")
(12, 12, 12, '    Again some spaces\n')
```

Some advice here. You can use a limited version of `split` for this or use more complex processing using `find`. We recommend using `split`, appropriate slicing operations, and perhaps `join`. This approach will save you time.

**To complete Checkpoint 1:** Show your code and output once you are finished.

### Checkpoint 2: Parsing some files

Copy your file from checkpoint 1 to a new file called `check2.py`. We will not use the function you have just implemented in this part, but it is good to keep it around for the next checkpoint. Make sure you save the program file into the same folder as the text files given in the ZIP folder.

This checkpoint is the second step to building your Easter egg hunt code. This is probably the most complex part of your lab. You will write a new function:

```
get_line(fname,parno,lineno)
```

which takes as input a file name and two numbers. Basically, the two numbers tell you which paragraph and which line your Easter egg hunt lies. Paragraphs are separated by any number of blank lines, containing nothing but the new line character and spaces.

You must open the given file, skip all the paragraphs up to the requested one (`parno`), skip all the lines until the given line (`lineno`), read and return the requested line. For example, suppose a file contains the following text:

```
I love deadlines. I love the whooshing
noise they make as they go by.


I refuse to answer
that question on the
grounds that
I don't know the answer.
```

If we are looking for paragraph 2 and line 3, your function should return the string `grounds that`. You should remove any trailing whitespace such as `'\n'`, but be sure to preserve leading whitespace. Refer back to lecture 13 and the str.rstrip() function if you are having problems with this part.

Write a program that asks the user a file number, a paragraph number and a line number. Then, call this function to find the line at this given location. For example, using the files given to you, here are some example runs:

```
Please enter the file number ==> 1
Please enter the paragraph number ==> 5
Please enter the line number ==> 4
import webbrowser/2/3/3

Please enter the file number ==> 2
Please enter the paragraph number ==> 3
Please enter the line number ==> 3
webbrowser.open("http://hackertyper.com/")/2/4/4

Please enter the file number ==> 2
Please enter the paragraph number ==> 4
```

```
Please enter the line number ==> 4
END/0/0/0
```

To accomplish this, think of using two variables: one for keeping track of paragraph number and the second one for the line number in that paragraph. Now, keep updating these numbers as you read a new line from the file.

**To complete Checkpoint 2,** show a TA or mentor your code once you have checked it for these test cases.

### Now for the Easter egg hunt!

If we look at the above examples, we can see the following. File 1, paragraph 5, line 4 contains the line: `import webbrowser/2/3/3`

This means two things:

- The first line of the program you are reading is: `import webbrowser`
- The next line of your program is at file `2.txt`, paragraph 3 and line 3.

The line at 2/3/3/ contains the content: `webbrowser.open("http://hackertyper.com/")`

This line points you to the next line, which is at 2/4/4. The line at this location has the content: `END/0/0/0`

This means, you are at the end of the program. By following these clues about where the next line is, you have read a program with the following lines:

```
import webbrowser
webbrowser.open("http://hackertyper.com/")
```

Your Easter egg hunt starts at a given line and continues to find the next line until you reach the end. All the lines you found are part of a small Python program.

### Checkpoint 3: Putting it all together

Now, we are going to put these together. Copy your code from the previous checkpoint to a file called `check3.py`.

As in checkpoint 2, you will ask from the user a starting point: file number, paragraph number and the line number.

You will first find the first line of the program. Then, you will construct the program using a WHILE loop, until you reach the line `END/0/0/0`. Write the lines you found in a file called `program.py`. When done, execute the recovered program. Try the following starting points:

```
1/5/4
8/43/4
3/87/7
```

You program must report failure if your parsing fails, which would happen if you have a bug in your code for counting paragraphs and lines or an incorrect starting point is given.

**To complete Checkpoint 3,** show your code to the TA who will test it with a given Easter egg.

### Things to think about

Well, you actually do not need to be told where the eggs are. There is an algorithm to recover them. Can you outline it or even better write it? Hint: it involves a type of while loop and very doable at your current level of programming.

There is a very easy algorithm that works for some eggs. But you need to get more sophisticated to find all of them. Remember: they are not all that well hidden.

Please post to Piazza your ideas of how to discover all the eggs, but do not post the locations. See if others can figure out where they are or how to find them. Some of these eggs are actually quite elaborate programs, so happy hunting!