Project 1

For Project 1, you will write a program that will display a rectangular grid of - and I. For example:

```
   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
0  - - - - - I I - - - - - - - - -
1  - - - - I - - I I - - - - - - -
2  - - - - I - - - - I I I I - -
3  - - - I - - - - - - - - I I I
4  - - I - - - - - - - - - - - -
5  - - - I - - - - - - - - I I -
6  - - - I - - I I I - - - I - I
7  - - - I - - I - - I - I - - -
8  - - - I I I I - - - I - - - -
9  - - - - - - - - - - - - - - -
```

The user will pick a (row, col) value and if the cell is occupied by -, it will fill it and the cells above, to the left, to the right, and below that are also occupied by – with a @. This should be done recursively for each cell (If your solution is not recursive, you will get very little credit for this project). You should not go out of bounds, try to fill diagonally, or fill a cell with an I (you can think of the I as a border). For example, if the user selects (7, 1), after the fill, the grid will look like this:

```
@ @ @ @ @ I I - - - - - - - - -
@ @ @ @ I - - I I - - - - - - -
@ @ @ @ I - - - - I I I I - - -
@ @ @ I - - - - - - - - I I I
@ @ I - - - - - - - - - - - - -
@ @ @ I - - - - - - - - I I -
@ @ @ I - - I I I - - - I @ I
@ @ @ I - - I @ @ I - I @ @ @
@ @ @ I I I I @ @ @ I @ @ @ @
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @
```

Here is what your program should do:

1. Prompt the user for a file until the user enters the name of a file that exists.

Sample run 0 (only part of the run is shown)

```
bash-4.1$ python proj1.py
Enter the name of the data file: fileNotHere.txt
Enter the name of the data file: iAmNotHere.txt
Enter the name of the data file:
```

2. Once you have a file, determine if it only consists of - and I. Of course there will be a newline character after the end of each row, but you should get rid of that before you populate your 2D list that holds the grid. If it consists of any other characters, your program should print an error message and end. For example, error1.txt (shown below) will be rejected because it has a ? in it on row 1 and a space in row 2.

Sample run 1

```
bash-4.1$ cat error1.txt
-----II--------
----I--II---?--
----I--- IIII--
bash-4.1$ python proj1.py
Enter the name of the data file: error1.txt
Your input has characters other than - and I
```

3. Assuming that it only has - and I, next make sure it is rectangular. If it is not, your program should print an error message and end. For example, error2.txt (shown below) will be rejected because it has 15 columns for row 1 and row 3, but 14 for row 2.

Sample run 2

```
bash-4.1$ cat error2.txt
-----II--------
----I--II-----
----I----IIII--
bash-4.1$ python proj1.py
Enter the name of the data file: error2.txt
Your input was not rectangular
```

4. Assuming you have a valid file, it should display the grid. Notice (see sample run 3) how the grid has spaces in between the - and I characters. The spaces were not read from the file, but added when displaying on the screen.

Next prompt for the row and column. Look at the grid at the start of the project document to understand the row/column numbering convention. Rows and columns start with zero, and row numbers increase as you go down and column numbers increase as you go to the right. Only proceed if you get a valid (row, column) pair. It is valid, if it is 2 integers that are not out of bounds.

Notice that in sample run 3, the pairs: `(hello, 5), (1, 5.0), (1, 5.1), (1, 15), (-1, 14)` are all considered invalid. Only after getting a valid pair, `(2, 7)` will it continue.

Sample run 3 (only part of the run is shown)

```
bash-4.1$ cat file1.txt
-----II--------
----I--II------
----I----IIII--
---I--------III
--I------------
---I--------II-
---I--III---I-I
---I--I--I-I---
---IIII---I----
---------------
bash-4.1$ python proj1.py
Enter the name of the data file: file1.txt
Valid input......will process
- - - - - I I - - - - - - - -
- - - - I - - I I - - - - - -
- - - - I - - - - I I I I - -
- - - I - - - - - - - - I I I
- - I - - - - - - - - - - - -
- - - I - - - - - - - - I I -
- - - I - - I I I - - - I - I
- - - I - - I - - I - I - - -
- - - I I I I - - - I - - - -
- - - - - - - - - - - - - - -
Enter the row number: hello
Enter the column number: 5
Enter the row number: 1
Enter the column number: 5.0
Enter the row number: 1
Enter the column number: 5.1
Enter the row number: 1
Enter the column number: 15
Enter the row number: -1
Enter the column number: 14
Enter the row number: 2
Enter the column number: 7
Print out step by step?  Enter Yes:
```

5. Once you have a valid (row, col) pair, give the user the option to see the recursive process one-step at a time or to just see the end product (The yes or no input by the user should not be case sensitive). In either case, you should print out what it looks like after it is done filling it with @.

Sample run 4

```
bash-4.1$ cat file2.txt
-----I-I------
----I---I-----
----I----IIII-
----I----IIII-
bash-4.1$ python proj1.py
Enter the name of the data file: file2.txt
Valid input......will process
- - - - - I - I - - - - - - -
- - - - I - - - I - - - - - -
- - - - I - - - - I I I I -
- - - - I - - - - I I I I -
Enter the row number: 2
Enter the column number: 6
Print out step by step?  Enter Yes: No
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: No
```

Sample run 5

```
bash-4.1$ python proj1.py
Enter the name of the data file: file2.txt
Valid input......will process
- - - - - I - I - - - - - -
- - - - I - - - I - - - - -
- - - - I - - - - I I I I -
- - - - I - - - - I I I I -
Enter the row number: 2
Enter the column number? 6
Print out step by step?  Enter Yes: yes

- - - - - I - I - - - - - -
- - - - I - - - I - - - - -
- - - - I - @ - - I I I I -
- - - - I - - - - I I I I -


- - - - - I - I - - - - - -
- - - - I - @ - I - - - - -
- - - - I - @ - - I I I I -
- - - - I - - - - I I I I -


- - - - - I @ I - - - - - -
- - - - I - @ - I - - - - -
- - - - I - @ - - I I I I -
- - - - I - - - - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ - I - - - - -
- - - - I - @ - - I I I I -
- - - - I - - - - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ - I - - - - -
- - - - I @ @ - - I I I I -
- - - - I - - - - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ - I - - - - -
- - - - I @ @ - - I I I I -
- - - - I @ - - - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ - I - - - - -
- - - - I @ @ - - I I I I -
- - - - I @ @ - - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ - I - - - - -
- - - - I @ @ - - I I I I -
- - - - I @ @ @ - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ - I - - - - -
- - - - I @ @ @ - I I I I -
- - - - I @ @ @ - I I I I -
```

```
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ - I I I I -
- - - - I @ @ @ - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ - I I I I -


- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -

***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: no
```

6. The program should keep looping until the user says to stop. In sample run 4 and 5, it stopped after one iteration, but in sample run 6 and 7 the user will iterate several times (The yes or no input by the user should not be case sensitive). Notice that if the user picks a cell that already has a `@` or `I`, it will just re-display the same grid. If they want to display step-by-step and they pick a cell that already has a `@` or `I`, there are no intermediary steps, so it ends up just displaying the same grid.

Sample run 6

```
bash-4.1$ python proj1.py
Enter the name of the data file: file2.txt
Valid input......will process
- - - - - I - I - - - - - -
- - - - I - - - I - - - - -
- - - - I - - - - I I I I -
- - - - I - - - - I I I I -
Enter the row number: 2
Enter the column number: 6
Print out step by step?  Enter Yes: no
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: yes

Enter the row number: 2
Enter the column number: 7
Print out step by step?  Enter Yes: no
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: YES
```

```
Enter the row number: 2
Enter the column number: 4
Print out step by step?  Enter Yes: no
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: yes

Enter the row number: 0
Enter the column number: 5
Print out step by step?  Enter Yes: yes
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: yes

Enter the row number: 0
Enter the column number: 6
Print out step by step?  Enter Yes: yEs
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: yes

Enter the row number: 0
Enter the column number: 0
Print out step by step?  Enter Yes: no
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
@ @ @ @ @ I @ I - - - - - -
@ @ @ @ I @ @ @ I - - - - -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -
Go again?  Enter Yes: no
```

Sample run 7

```
bash-4.1$ python proj1.py
Enter the name of the data file: file2.txt
Valid input......will process
- - - - - I - I - - - - - -
- - - - I - - - I - - - - -
- - - - I - - - - I I I I -
- - - - I - - - - I I I I -
Enter the row number: 2
Enter the column number: 5
Print out step by step?  Enter Yes: no
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
- - - - - I @ I - - - - - -
- - - - I @ @ @ I - - - - -
- - - - I @ @ @ @ I I I I -
- - - - I @ @ @ @ I I I I -
Go again?  Enter Yes: yes

Enter the row number: 1
Enter the column number: 1
Print out step by step?  Enter Yes: no
***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
@ @ @ @ @ I @ I - - - - - -
@ @ @ @ I @ @ @ I - - - - -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -
Go again?  Enter Yes: yes
```

```
Enter the row number: 1
Enter the column number: 12
Print out step by step?  Enter Yes: yes

@ @ @ @ @ I @ I - - - - - -
@ @ @ @ I @ @ @ I - - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I - - - - @ -
@ @ @ @ I @ @ @ I - - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I - - - @ @ -
@ @ @ @ I @ @ @ I - - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I - - @ @ @ -
@ @ @ @ I @ @ @ I - - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I - @ @ @ @ -
@ @ @ @ I @ @ @ I - - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ -
@ @ @ @ I @ @ @ I - - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ -
@ @ @ @ I @ @ @ I @ - - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ -
@ @ @ @ I @ @ @ I @ @ - @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ -
@ @ @ @ I @ @ @ I @ @ @ @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ @
@ @ @ @ I @ @ @ I @ @ @ @ -
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ @
@ @ @ @ I @ @ @ I @ @ @ @ @
@ @ @ @ I @ @ @ @ I I I I -
@ @ @ @ I @ @ @ @ I I I I -
```

```
@ @ @ @ @ I @ I @ @ @ @ @ @
@ @ @ @ I @ @ @ I @ @ @ @ @
@ @ @ @ I @ @ @ @ I I I I @
@ @ @ @ I @ @ @ @ I I I I -


@ @ @ @ @ I @ I @ @ @ @ @ @
@ @ @ @ I @ @ @ I @ @ @ @ @
@ @ @ @ I @ @ @ @ I I I I @
@ @ @ @ I @ @ @ @ I I I I @

***THIS IS WHAT IT LOOKS LIKE AFTER PROCESSING YOUR ROW AND COL***
@ @ @ @ @ I @ I @ @ @ @ @ @
@ @ @ @ I @ @ @ I @ @ @ @ @
@ @ @ @ I @ @ @ @ I I I I @
@ @ @ @ I @ @ @ @ I I I I @
Go again?  Enter Yes: no
```

You should use top-down design to create this project. You need to have several functions (you decide how many) in addition to main. No function can be more than 20 lines of code. Main should be 35 lines or less. As before, comments and function header does not count as line numbers. You **must** have the following function (should be 20 lines or less):

`def fillWithAT(grid, theRow, theCol, display):`

This is the function that will fill the cell with the – with @ and recursively do that for the cell above, to the left, to the right, and below (in that order). The arguments passed in are the grid, the row and col, and a Boolean value for display. If it is `True`, you will display step by step.

As always:

Use the sample runs as a guide for developing your program and for the format of the prompts and the outputs. Don't add anything extra and create your program to exactly match the specifications given above and in the sample runs. In the sample runs, the user input is shown in bold so you can easily distinguish input from program generated text (it does not have to show up as bold in your program).

IMPORTANT: Name your file proj1.py. Only submit proj1.py. Don't submit your sample input text file (we will make our own).

VERY IMPORTANT: The whole point of the project is recursion. That means that if you have a non-recursive solution that matches the output above 100%, you will still get very little credit for this project.

When you've finished your project, use the submit command to submit the file.

You must be logged into your account and you must be in the same directory as the file you're trying to submit.

At the Linux prompt, type

```
submit cs201 Proj1 proj1.py
```

After entering the submit command shown above, you should get a confirmation that submit worked correctly:

```
Submitting proj1.py...OK
```

If not, check your spelling and that you have included each of the required parts and try again.

You can check your submission by entering:

```
submitls cs201 Proj1
```

You should see the name of the file that you just submitted, in this case proj1.py