Function Name: `sk8erBoi`

Inputs:

1. (*char*) Filename with boy's love letter
2. (*char*) Filename with girl's rejection letter

File Outputs:

1. A .txt file with combined letters

Topics: (*line-by-line reading*), (*line-by-line writing*)

Background:

You've found Avril Lavigne's secret stash of letters! Each love letter has a corresponding rejection letter. As an avid Avril Lavigne fan, you must compile the matching letters in order to uncover the full stories.

Function Description:

Given two filenames with the same number of lines, print alternating lines starting from the first line of the first file to a new file named `'<filename1>&<filename2>.txt'.`

Example:

`love.txt →`

```
1| I love hearing you laugh!
2| You look so beautiful in your purple dress.
3| Will you go out with me?
```

`rejection.txt →`

```
1| You're boring.
2| You're ugly.
3| nah.
```

```
sk8erBoi('love.txt','rejection.txt');
```

`love&rejection.txt →`

```
1| I love hearing you laugh!
2| You're boring.
3| You look so beautiful in your purple dress.
4| You're ugly.
5| Will you go out with me?
6| nah.
7|
```

Notes:

- The two files will always have the same number of lines.
- Your output should have an extra newline character at the end

Function Name: `girlfriend`

Inputs:

1. (*char*) Filename with mixed up song lyrics

File Outputs:

1. A .txt file with the lines correctly ordered.

Topics: (*cell arrays*), (*sorting*)

Background:

You're writing Avril's next chart topper! Unfortunately, your bae is distracting you so much that you're writing all the heart wrenching lines out of order!

Function Description:

Given the filename of a .txt file with lines of the form '<new line number> <song lyrics>', correctly order the file lines and print those lines to a new file named `'<filename>_reordered.txt'`. The line numbers should be deleted from the lines when they are printed to the new text file. An extra newline at the end of the output text file is required.

Example:

`lyrics.txt` →

```
1| 2 No way, no way, I think you need a new one
2| 3 Hey hey, you you, I could be your girlfriend
3| 1 Hey hey, you you, I don't like your girlfriend
```

```
girlfriend('lyrics.txt');
```

`lyrics_reordered.txt` →

```
1| Hey hey, you you, I don't like your girlfriend
2| No way, no way, I think you need a new one
3| Hey hey, you you, I could be your girlfriend
4|
```

Notes:

- There will always be exactly one space between the correct line number and the song lyrics

Hints:

- The `sort` function works on cells containing data type char, but not cells containing data type double

Function Name: `helloKitty`

Inputs:

1. A filename of song lyrics
2. (*cell*) A 1XN cell array containing words

File Outputs:

1. A .txt file of the fixed song

Topics: (*word-by-word reading*), (*word-by-word writing*)

Background:

During a performance of her hit 2013 Japanese culture influenced song "Hello Kitty", the robot that replaced Avril Lavigne malfunctions. She randomly says `'BEEP'` instead of her song lyrics, and it's your job to fix it!

Function Description:

Given a text file(`'<filename>.txt'`) with song lyrics, and a 1xN cell array of words, write a new text file named `'<filename>_fixed.txt'` you replace each instance of `'BEEP'` in the original text file with the next word of the cell array. There will always be the same number of `'BEEP'`s and words in the cell array. An extra newline at the end of the output text file is required.

Example:

`song.txt` →

```
1| Chill BEEP what you yelling BEEP
2| Lay back, it's all been done before
3| And BEEP you could only let it be, you will see
```

```
words = {'out','for','if'};
helloKitty('song.txt',words);
```

`song_fixed.txt` →

```
1| Chill out what you yelling for
2| Lay back, it's all been done before
3| And if you could only let it be, you will see
4|
```

Notes:

- There will be at least one `'BEEP'` in every input file
- There can be multiple instances of `'BEEP'` on one line
- `'BEEP'` will never be preceded nor followed by punctuation.

Function Name: `whatTheHeck`

Inputs:

1. (*char*) The filename of the text file that you want to count the words in

File Outputs:

1. A .txt file containing the words and their counts

Topics: (*word-by-word reading*), (*cell arrays*)

Background:

To see if Avril Lavigne is a robot programmed with a limited vocabulary, you decide to write a function that counts the number of times words appear in her songs. You're on your knees, begging please, count the words! But honestly, I just need to be a little crazy.

Function Description:

Given a `.txt` file, count the number of times each unique word appears and print the words and their counts to a new file named `'<filename>_counts.txt'`.

Example:

`twinkle.txt` →

```
1| twinkle 1738 twinKLe## little STAR
```

```
whatTheHeck('twinkle.txt')
```

`twinkle_counts.txt` →

```
1| twinkle 2
2| little 1
3| star 1
4|
```

Notes:

- The words in the output text file should be lowercase and in order of their first appearance in the input text file.
- Remove all characters but spaces and letters before looking for a word.
- Your function should **not** find words within words (i.e. `'loop'` in `'loops'` does not count).
- Your function should be case insensitive (i.e. `'Matlab'` and `'MATLAB'` count as the same word).
- The last character in the output file should be a newline character.
- If you want to test different files, or investigate a specific piece of work, check out archive.org and search for your text of choice.

Function Name: `complicated`

Inputs:

1. (*char*) The filename of a file containing random characters
2. (*char*) The filename of a file containing directions

Outputs:

1. (*char*) The string of characters found along the path

Topics: (*multiple reading of single file*) -or- (*chain cell indexing*)

Background:

♪♫♪♫ Why'd Homework Team have to go and make this problem so complicated? ♫♪♫♪

Function Description:

Given the filename of a text file containing a corrupted song file, start at the top left corner and follow the directions in the text file given by the second input. In the directions file, each line will be of the format '`<direction> <# of steps>`'. Output the characters you travel through as a character vector to decode the message in the file!

Example:

The green character is the starting point, the yellow is the path, and the red is the end.

`song.txt` →

```
1| MATsk
2| $TL9co$l
3| E%AB r
```

`directions.txt` →

```
1| right 2
2| down 2
3| right 3
4| up 1
5| left 1
6| up 1
7| left 1
```

```
str = complicated('song.txt', 'directions.txt')
str → 'MATLAB rocks'
```

Notes:

- The lines of the first text file will not be the same length.
- The cursor will always start on the first character in the file.
- The path will never go out of bounds of the file.