Exam 3B

Comp 123

- You may use the computer, and any resources on Moodle, Piazza, the Interactive Python textbook, your online journal, or the Python.org documentation pages. You may also use any paper resources you like, including the Guzdial and Ericson textbook.
- You **may and should** ask me for clarifications, or hints.
- You have 30 minutes to complete this test.

1. (10 pts) Use the `imageTools` module and write a function called `horizLines`, saving it in `exam3BCode.py`. This function should take a picture object, a color object, and a distance (an integer) as its inputs. This function should **make a copy** of the input picture, and then change certain rows of the picture to be the input color. Row 0 should be changed to the color, and then it should skip by the input distance before changing another row. The function should **return** the new picture at the end.

   Hints:
   - This problem can be done either by looping over all pixels as in Chapter 3, or by nested `for` loops as in Chapter 4.
   - The rows that should be changed have y values that are a multiple of the input distance. For instance, if the input distance is 12, then rows 0, 12, 24, 36, and so forth should be changed.
   - If you need to check whether something is a multiple, remember the remainder operator: If z is a multiple of d, then the remainder of z divided by d is zero.

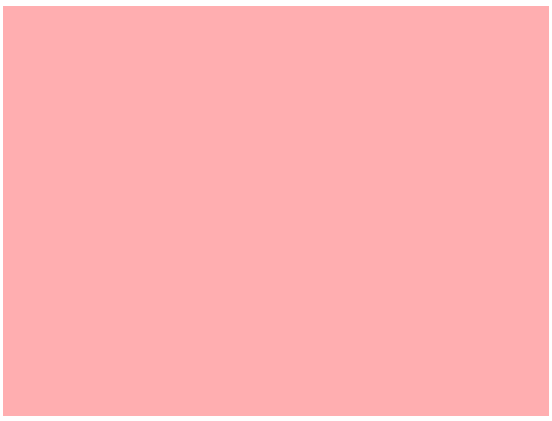
   Below are the sample calls that are in your `exam3BCode.py` file, and what they should produce (zoom in to see the dots).

```
pic1 = makePicture("blueMotorcycle.jpg")
np1 = horizLines(pic1, red, 20)
np2 = horizLines(pic1, cyan, 50)
np3 = horizLines(pic1, green, 5)
np4 = horizLines(pic1, pink, 1)
```

np1	np2

np3

np4



2. In the `exam3BCode.py` file, you will find a function that does string operations: `blankBadWords`. This function takes two inputs: a string and a list of strings. The function goes through the input string, and looks for any occurrences of strings from the list of strings. It replaces every occurrence with the same number of Xs, blanking out the “bad words” from the list. It first replaces the first “bad word” from the list, and then starts on the second one: if the bad words might overlap, then replacing one might change whether another matches (see examples below).

   Below are some sample calls to `blankBadWords`, and what the correct result should be:

```
>>> blankBadWords("A man, a plan, a canal: Panama!", ['an', 'ma'])
A mXX, a plXX, a cXXal: PXXaXX!
>>> blankBadWords("A man, a plan, a canal: Panama!", ['ma', 'an'])
A XXn, a plXX, a cXXal: PXXaXX!
>>> blankBadWords("A man, a plan, a canal: Panama!", [])
A man, a plan, a canal: Panama!
>>> blankBadWords("The quick brown fox jumps over the lazy dog",
```

```
                  ['a', 'e', 'i', 'o', 'u'])
ThX qXXck brXwn fXx jXmps XvXr thX lXzy dXg
```

There are **three bugs** in this function. Find them, label the lines they are on, and correct them!