CS101 Homework #2

Selfie Decoration

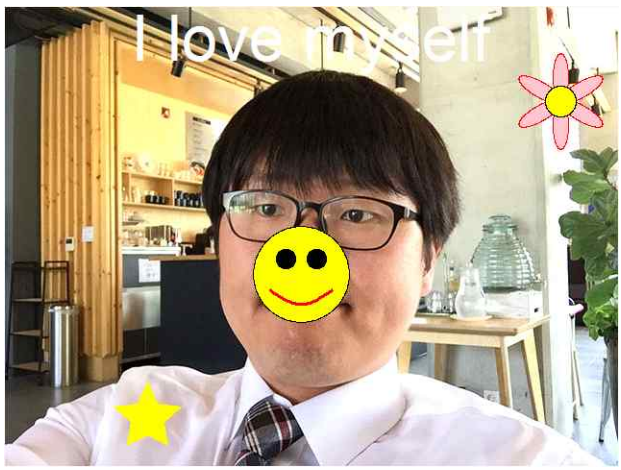
Due Date: Sunday, November 1, 2015 (Until 23:59)
Delayed due date: Wednesday, November 4, 2015 (Until 23:59)

Please read the homework description carefully and make sure that your program meets all the requirements stated. The homework is an individual task. You can discuss the problem with your friends but you should not program together. **You will get an F on the entire course if your homework includes any plagiarism.**

Goal

In this homework, you need to implement a Python program by which you can decorate your selfie (or any photo). Please see the pictures below as an example. By using the program, the brightness of the photo has been increased by 20%, and three graphical marks and a text have been added.

Before



After

In this homework, you are required to implement two filters: a gray-scale filter and a light-intensity filter. In addition, you need to design and implement three graphical marks to put on.

cs1graphics_HW2 module

Unfortunately, the 'cs1graphics' module does not include enough functionality for this homework. Thus, we have made an improved version of the module, named 'cs1graphics_HW2'. To use the new module, you can either copy the *cs1graphics_HW2.py* file to *C:₩Python27₩Lib₩site-packages*, or just put the file to the same directory where your source code exists.

The major update that has been made on the module is to add four new functions to the Image class. The following is the specification of the functions. Actually, the functions are quite similar to the functions in the 'cs1media' module, except for the 'commit()' function.

size()	Parameters	None
	Behavior	Get the size of the current image and return it.
	Return value	A tuple with 2 integers, which means (width, height).
get(x, y)	Parameters	x: x-coordinate of the point y: y-coordinate of the point
	Behavior	Return the color of (x, y)
	Return value	A tuple with 3 integers, which represent the RGB color values of the point (x, y).
set(x, y, color)	Parameters	x: x-coordinate of the point y: y-coordinate of the point color: a tuple with 3 integers, (R, G, B).
	Behavior	Set the color of the point (x, y) to the color specified in the parameter. Note that this function set the color, but do not re-draw the image. Use the commit() function to apply the changes. You can make a set of changes using the set() function, then call the commit() function once to apply all the changes.
	Return value	None.
commit()	Parameters	None.
	Behavior	Refresh the image. Apply the changes made by using the set() function.
	Return value	None.

The following is an example program that loads an image, and paints the bottom-right point with the color at the top-left point by using the functions defined above.

```
from cs1graphics_HW2 import *

img = Image('Changdae.jpg')
width, height = img.size()
canvas = Canvas(width, height)
canvas.add(img)
r, g, b = img.get(0, 0)
img.set(width-1, height-1, (r, g, b))
img.commit()
```

In addition, the checkColorName() function has been added to the new module. This function takes the name of a color and checks if the name is a valid color name or not. You do not need to use the checkColorName() function directly in this homework. However, you can use the 'color_input()' function in the template code to get a valid color name from a user.

Requirement

A. You should implement the six functions in the template code. The details of the functions are explained below.

1) Note that we provide some implementations in the template. Please read the code carefully and try to use the functions for blurring an image, putting a text message on an image, etc.

B. Your source code should contain appropriate comments.

1) The title of the program and the author information must be written at the head of your source code.
2) A brief step-by-step description of the algorithm should be included at appropriate positions.
3) It is highly recommended to add comments to improve the readability of your source code. If your source code contains only a few comments, you will get some penalties.

C. You must submit a report. There is no strict format, but only doc, docx, and pdf files are allowed for your report. The report should include the followings:

1) Introduction to your program.
2) Explanation about your implementation. Focus on what you need to explain when someone brings and uses your code to improve his/her program.
3) Your selfies or other photos decorated by using your program. You should show the results of using all the features that you implemented. You can use multiple photos to demonstrate the features.
4) Explanation about additional functionalities that you like to add to this program, and how those functionalities can be implemented. (A rough implementation strategy is okay.)
5) Describe what you have learned and felt while doing this homework.

D. There are two restrictions. Note that you will not be able to get scores if you do not keep the followings.

1) You MUST follow the function name, the order of parameters for each function, and the type of parameters for each function. Your code will be scored at the function level. If you do not follow, your program cannot be scored. Simply, do not modify the definition lines.
2) You MUST not modify the main() function. If you add any line in the main() function, the lines will not be considered during scoring. If you think that main() has errors or lacks some functionality, please let the TAs know. Outside the main() function, you can add any number of functions and global variables.

The followings are the functions that you need to implement.

1. def gray_scale(img)

Parameters	**img:** an image object
Return value	None

Modify the image to a gray-scale image. 'Gray scale' means that in any point, its R, G, and B values are same. That is, (0,0,0) and (55, 55, 55) are gray-scale colors, but (5, 10, 15) is not.
You should use the luminance value to get appropriate R, G, B values. For the luminance value, refer to the photo processing lecture.
Note that this function should not affect other graphical marks or texts in the canvas.

Before

After

2. def make_lighter(img, factor_percent)

Parameters	**img:** an image object **factor_percent**: the percentage of how much light intensity will be increased
Return value	None

Modify the image to be brighter. If factor_percent = 20, it means 20% of light intensity will be increased. That is, if R value is 150 and factor_percent = 20, R value will be set as 180 (=150 X 1.20).

Note that this function should not affect other graphical marks or texts in the canvas.

Before

After
factor_percent = 40

3. def put_smile_mark(x_pos, y_pos, size)

Parameters	**x_pos**: an x-coordinate of the center of the smile mark. **y_pos**: an y-coordinate of the center of the smile mark. **size**: the size of the smile mark, that is, the diameter of the smile mark.
Return value	None.

Draw a smile mark centered at (*x_pos*, *y_pos*) with *size* as its diameter.

Your program should meet the following conditions. To help you to meet those conditions, the template code includes a guide box for each function. The guide box is a square box with the size, *size*, and centered at (*x_pos*, *y_pos*). The guide box can be drawn by using the following code.

```
guide_box = Rectangle(size, size, Point(x_pos, y_pos))
```

Note that you can see the guide box in the 'test mode'. The 'test mode' will be explained later.

The conditions to meet:

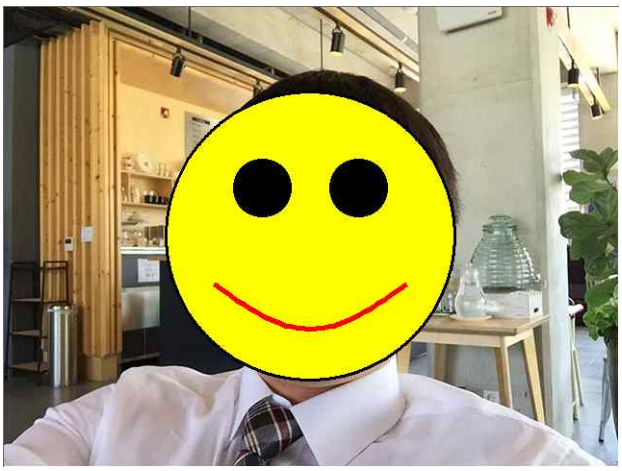
1) One circle for the face, filled with yellow color.
2) Two circles for the eyes. One should be in the upper-left quadrant, and another should be in the upper-right quadrant. The color of the eyes should be different from the colors of the face and the mouth.
3) One path or spline for the mouth. If you use a path, at least one bending point is necessary. The mouth should be at the bottom half of the guide box. The color of the mouth should be different from the colors of the face and the eyes.

Regarding to the size of the smile mark, you should satisfy the following conditions::

1) Every part of your mark should be shown inside the guide box.
2) At least one point of your mark should meet a side of the guide box.

You should be careful with setting the depth of each object on the image. You can use the global variable, *depth*, which is defined in the first part of the template code.

Before



After
x_pos = 320, y_pos = 240, size = 300

4. def put_star_mark(x_pos, y_pos, size)

Parameters	**x_pos**: an x-coordinate of the center of the star mark. **y_pos**: an y-coordinate of the center of the star mark. **size**: the size of the star mark.
Return value	None.

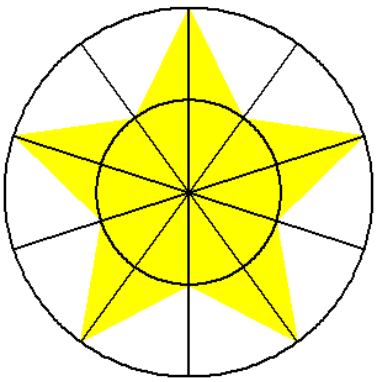
Draw a star mark centered at (*x_pos*, *y_pos*).
Similar to the smile mark, the size of the star mark should be relative to the guide box, which is a square box with the size, *size*, and centered at (*x_pos*, *y_pos*).

You should draw a five-pointed star, as the following example.

Before

After
x_pos = 320, y_pos = 240, size = 300

The only condition that you should satisfy is that the star is a five-pointed one. However, we provide you a hint. See the right figure. There are two circles. The diameter of the outer circle is *size*, and the diameter of the inner one is *size / 2*. In addition , there are five lines. The lines evenly divide the circles. The adjacent two lines make an angle of 36 degrees. Now, except for the center, the crossing points make a five-pointed star. Don't worry about how to get the crossing points. You can use the sine and cosine values multiplied by *size* or *size / 2* for each point.



Regarding to the size of the star mark, you should meet the following conditions:
1) Every part of your mark should be shown inside the guide box.
2) At least one point of your mark should meet a side of the guide box.

Again, you should be careful with setting the depth of the objects.

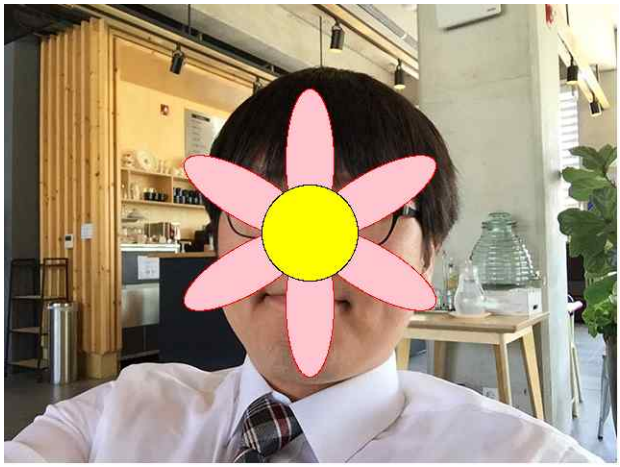
5. def put_my_mark(x_pos, y_pos, size)

Parameters	**x_pos**: an x-axis position of the center of mark. **y_pos**: an y-axis position of the center of mark. **size**: the size of the mark.
Return value	None.

Draw your own mark centered at (*x_pos*, *y_pos*).
Similar to the other marks, the size should be defined to be relative to the guide box, which is a square box with the size, *size*, and centered at (*x_pos*, *y_pos*).

For example, we can draw a flower mark as follows.

Before



After
x_pos = 320, y_pos = 240, size = 300

The mark that you define should consist of at least 4 objects.
The size constraints are the same as above.
1) Every part of your mark should be shown inside the guide box.
2) At least one point of your mark should meet a side of the guide box.

6. def range_input(prompt, low, high)

Parameters	**prompt**: a text message to show in Python shell. **low**: the lowest acceptable value. **high**: the highest acceptable value.
Return value	An integer value of a user's input which is greater than or equal to *low* and less than or equal to *high*.

This function is to help users enter a valid integer value, which should be in a certain value range, when your program requires the value as an input. Firstly, the function shows the text message with the value range of acceptable integer values. Then, it asks for an input to the user. If the user enters a valid integer value between *low* and *high*, the function returns the value. Otherwise, it asks the user again for a new value until the user enters a valid one.
You should use this function whenever you want to ask users an integer input. The main() function that we provide in the template code also uses this function to get user inputs. For example, when a user selects a menu, the main() function does not use the raw_input() function. Instead, it calls the range_input() function with low = 1 and high = 9. This helps the user to select a valid menu item from available menus.

Here are examples of using the function. Please note that you should implement your program to generate similar messages.

```
Enter an x-coordinate (0 ~ 639) -1
Error! Enter a number between 0 and 639
Enter a x-coordinate (0 ~ 639) 650
Error! Enter a number between 0 and 639
Enter a x-coordinate (0 ~ 639) 320
```

Test mode

While working on this homework, you may need to run your program many times. Entering inputs such as the filename of a photo, a menu number, and coordinate values for each run might be a hassle for you. Thus, we provide the 'test mode' of the program for your convenience.
In the test mode, a default photo will be selected automatically, and some functions will be run with sample parameters. With the test mode, you can test your program without entering all the inputs repeatedly. In addition, the guide boxes are shown in the test mode. This will help you to meet the conditions that are explained previously.

The below is the part of the template source code.

```
# test mode flag. When you submit or get the final results, set this False.
test_mode = False
```

If the variable set to True, the program runs as the test mode.

You can use the test mode while doing your homework. However, you should check your program with other inputs, that is, other photos, other parameters, and graphical marks in various positions and sizes.

Submission

You need to submit the followings:

- The program file: "HW2_yourid.py"
  (e.g.) HW2_20151234.py
- The homework report: "HW2_yourid.doc" or "HW2_yourid.docx" or "HW2_yourid.pdf"
  (e.g.) HW2_20131234.doc, HW2_20131234.docx, HW2_20131234.pdf

You MUST compress the source code and the report together into "HW2_yourid.zip" (e.g., HW2_20151234.zip) and submit the compressed file via the webpage for homework submission.

If you do not follow the above submission policy, you will get penalty.

Late submissions will get a 5% penalty per day from November 2nd to 4th. After November 4th, the server will be closed and no more submissions will be accepted. Please try to submit your homework before the deadline.