Honors Computer Science Python
Mr. Clausen
Program 7A, 7B

PROGRAM 7A Turtle Graphics Animation (100 points)

Here is the overview of the program. Use functions to draw a minimum of two background scenes. These can be nature scenes, or city scenes with **a lot of details** in either choice. Since I am asking for nature or city scenes, no "abstract art" is permissible, you have to draw something recognizable as a nature or city scene. You may use some algorithmic drawing to add realism, but only as a small percentage of your code. For example, you may use fractals to draw better looking tree branches or blades of grass. You can use any of the commands that are in the PowerPoint presentation or on the Turtle Graphics Documentation web page. **Don't make the majority of your scene the same Geometric shape (so, no "space" scenes where most of the shapes are circles, for example)**. Start by going to the Resource folder for this class "HnrCSPythonFiles", and find the folder "Turtle Graphics". Copy the two files "TurtleGraphicsTemplate.py", and "turtle.cfg" to your "S:" directory. Open the template and use this to create your artwork. Do not change any of the screen sizes in either file. **Make sure to change the comments in the template to include your information.**
After you create the two or more background scenes, use the PrintScreen key on your keyboard to do screen captures, edit the pictures to crop, and resize them to fit the window when you load them in (usually somewhere around 1000 by 500 or 600) and save them as transparent GIF files.
Next, comment out the function calls to all the functions that draw Scene1, Scene2, etc. (do NOT delete the functions) and load each background picture when necessary. Instantiate a minimum of two turtles and assign a transparent gif files to each turtle. One of the turtle shapes must be drawn in a paint program, while the other can be downloaded. Move these turtles across the screen **simultaneously** for your animation (this program is like program L5 in the Intro to Programming class).

1) Before you do anything else, create a folder named LastNameFirstNameP7A in your "S:" directory. Then go to the Resource folder ("R: directory") for this class "HnrCSPythonFiles", and find the folder "Turtle Graphics". Copy the two files "TurtleGraphicsTemplate.py", and "turtle.cfg" into the folder "LastNameFirstNameP7A" in your "S:" directory. As you work through this project, make sure **EVERYTHING** you need to run this program is saved into this folder.

2) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

3) Leave a blank line after the DocString.

4) Use the following import statements:
   from turtle import *
   AND

import random (if you wish to randomize the speeds of the turtles)

5) Leave a blank line after the import statement.

6) Type a one line comment of all equal signs.
#==========

7) Define the **main** function. The main function should declare and initialize all of the variables (except any variables that are better left as local variables within other functions) and call other functions. The last lines of code in the main function should be:
screen.delay(1000)
exitonclick( )
input("Press enter to close the turtle graphics window")
bye( )

8) Initialize all of the variables that will be passed from the main function and instantiate all of the objects that are to be used in this program **in the main function** and call the functions from the main function (for example, drawSceneOne( ), drawSceneTwo( ), animateSceneOne( ), etc.). Have these functions call other functions. For example, drawSceneOne( ) should call all of the other functions used to draw the first scene. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal (float) numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program (except for variables that are local variables in other functions). Make sure that you use descriptive identifiers for all of your object and function names. **You can use "t1", "t2", "t3", etc. for the Turtle objects.**

9) After the main function, type a one line comment of subtraction signs. **Include this comment after each function as well.**
#----------

10) Define the function displayMyInfo. Inside this function use print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.** Include a comment line of subtraction signs after the function. Call this function from the main function and run your program to see if it works without errors.

11) **For all function definitions from here to the end of the program, remember to include a DocString at the beginning of each function with a brief description of what the function does (draws).**

12) Now for the fun part. Create a separate function for each part of each scene. For instance, if you draw a mountain scene, you will need separate functions to draw the mountain, the sky, trees, etc. If the mountain has a lot of detail, you should also have separate functions for each part of the mountain. You must use functions throughout this program.

13) For some guidelines as what to include in your drawing, you must use color in your drawing. **If you use any custom colors based upon RGB ratios or hexadecimal numbers, you must include a comment after the color to tell me what color it is.** You must also FILL in all closed

shapes.

14) After you create the two or more background scenes, use the PrintScreen key on your keyboard to do screen captures, edit the pictures to crop, and resize them to fit the window when you load them in (usually somewhere around 1000 by 500 or 600) and save them as transparent GIF files.

15) Next, comment out the function calls to all the functions that draw Scene1, Scene2, etc. (do NOT delete the functions) and load each background picture when necessary.

16) Instantiate a minimum of two turtles and assign transparent gif files to each turtle. One of the turtle shapes must be hand drawn in a paint program, while the other(s) can be downloaded.

17) Move the turtles across the screen simultaneously (or at least give the appearance of simultaneous movement) for your animation (this program is like program L5 in the Intro to Programming class). Name these functions animate1( ), animate2( ), etc.
Inside the animation functions, load the background scene using the screen.bgpic("scene1.gif") command. Hide each turtle that you are using in each scene and set the shape you wish to move across the screen using the t.shape("myTurtle1.gif") command. Move each turtle to their starting positions and animate them.

18) On a line by itself after all of the functions type: **main()** #You will need this line of code to call the main() function, or your program will not run.

19) Type two comment lines of equal signs to indicate that this is the end of the program.
#==========
#==========

20) Save your program as LastNameFirstNameP7A.py.

21) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then copy and paste your folder for program 7A in the "T" network mapping, in the Program 7A folder.

PROGRAM 7B Image Processing (30 points)

Here is the overview of the program. We are writing an image processing program that will take a .gif file (smokey.gif – the cat picture that came with our textbook) and do some simple image processing similar to a graphics program like Photoshop. **Write code to flip the picture vertically, flip the picture horizontally, rotate the picture 90° clockwise, and rotate the picture 90° counterclockwise**. Note: the picture, smokey.gif is not a square, but rectangular in its dimensions. Make sure that your source code and the picture "smokey.gif" are in the same folder. You can find the picture of Smokey.gif in the folder path: \Resource\HnrCSPythonFiles\Ch 7 Image Files. There is a sample program in the same folder that copies the picture pixel by pixel and saves the copy. Feel free to use this as a reference, since you will be changing a copy of the picture and not the original image.
You need to display the original picture and the processed **copy** of the picture after performing each effect. After processing the picture, save the processed copy using a different file name. For example, after rotating the picture clockwise 90°, save the picture as:
"YourLastNameYourFirstNameSmokeyRotate90CW.gif".

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) Use the following import statement: **from images import Image**
(Note: this will only allow us to do image processing on .gif files and not any other format.)

4) Leave a blank line after the import statement.

5) Type a one line comment of all equal signs.
#==========

6) Define the **main** function. The main function should declare and initialize all of the variables (except any variables that are better left as local variables within other functions) and call other functions. The last line of code in the main function should be:
input("Press enter to quit the program")

7) Initialize all of the variables that are to be used in this program **in the main function** and call all of the other functions from the main function. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal (float) numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program.

8) After the main function, type a one line comment of subtraction signs. **Include this comment after each function as well.**
#----------

9) Define the function displayMyInfo. Inside this function use print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.** Include a comment line of subtraction signs after the function. Call this function from the main function and run your program to see if it works without errors.

10) **For all function definitions from here to the end of the program, remember to include a DocString at the beginning of each function with a brief description of what the function does. (This program does not lend itself well to type contracts and sample function calls.) DO NOT NAME ANY FUNCTIONS WITH NAMES LIKE: menuChoice1, etc. THE NAME OF THE FUNCTION SHOULD DESCRIBE WHAT THE FUNCTION DOES, NOT HOW IT RELATES TO THE MENU.**

11) Define the function "menu" that displays the menu (as pictured below) and asks the user to enter their choice and returns this choice to the main function. Upper and lower case values should work for the menu choice to quit. Pass the variable representing the user's choice, and anything else you think you may need to this function when calling it from the main function.
Next, inside the menu function type all of the "if" "elif" choices for the program. For now, just place a print statement in each choice saying what choice it represents. **As you define each function, call the function from the menu function (if you choose the four function approach).** Run your program to see if this works without errors.

```
 Image Processing
1. Flip Horizontal
2. Flip Vertical
3. Rotate 90 degrees Clock Wise
4. Rotate 90 degrees Counter-Clock Wise
Q. Quit the program
Enter your choice
```

12) In the main( ) function, create a "while loop" that will quit when the user enters either a lowercase 'q' or upper case 'Q' and keep running otherwise.
Now call the "menu" function from the main function **inside the "while loop"** and run your program to see if it works without errors, and will quit when the user enters either an uppercase or lowercase letter 'Q'.

13) Follow the sample program that displays the original picture and prints the message: "Close the image window to continue" (This is in the "R:" directory). After closing the original picture, display the modified picture. After closing the modified picture, return to the menu (so you will need a loop with the menu).

14) You need to display the original picture and the processed **copy** of the picture after performing each menu choice effect. After processing the picture, save the processed copy using a different file name. For example, after rotating the picture clockwise 90°, save the picture as: "LastNameFirstNameSmokeyRotate90CW.gif".

15) After the menu function, you could define four functions, each one corresponding the menu choice, or you could accomplish all four tasks with one function while changing parameters based on the menu choice. If you choose the one function approach, the "if, elif statements" should be in this function.

16) On a line by itself after all of the functions type: **main()** #You will need this line of code to call the main() function, or your program will not run.

17) Type two comment lines of equal signs to indicate that this is the end of the program.
#==========
#==========

18) Save your program as LastNameFirstNameP7B.py.

19) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 7B folder.