Assignment 1, Part B: Personalised Graffiti Tag

(5%, due 5:00pm Wednesday, September 13th)

Overview

This is the second part of a two-part assignment. This part is worth 5% of your final grade for IFB104. Part A which preceded it was worth 20%. This part is intended as a last-minute extension to the assignment, thereby testing the maintainability of your code from Part A and your ability to work under time presssure. If you have a neat, clear solution to Part A you will find completing Part B easy. For the whole assignment you will submit only one file, containing your combined solution to both Parts A and B, and you will receive one grade for the whole 25% assignment.

Motivation

One of the most common tasks in "Building IT Systems" is modifying some existing code. In practice, computer programs are written only once but are subsequently modified and extended *many* times during their operational lifetime. Code changes may be required in response to internal factors, such as the need to correct design flaws or coding errors, or external factors, such as changes in consumer requirements.

This task requires you to extend your solution to Part A of the assignment by adding an additional feature. It tests:

- Your ability to work under time pressure; and
- The quality and clarity of your code for Part A, because a well-written solution to Part A will make completing this part of the assignment much easier.

Goal

In Part A of this assignment you were required to create code that simulated a billposter pasting paper sheets onto a billboard. However, one of the hazards associated with advertising billboards is that they are often vandalised with graffiti tags, as shown above and below. Therefore, in Part B of this assignment you will simulate this real-life scenario by "tagging" your own billboard with a graffito[1].

To complete this task you must modify your solution to Part A so that it draws a graffiti "tag" on any billboard which has an '`X`' at the start of the data set used to create it. The graffito must cover the whole billboard and must be a tag consisting of **your initials**, at least two. You can include additional graffiti if desired, but **your initials must be the dominate part**. The initials **must be drawn** with the "turtle" cursor, not printed, to look as if they were spray painted across the billboard. You may **not** use Turtle's `write` function to print the initials.

You will complete this part of the assignment by extending your code for Part A. No additional Python template file is supplied for this part. Your program must work for all of the data sets in the supplied Python file, and **any other similar data sets in the same format**.

Illustrative example

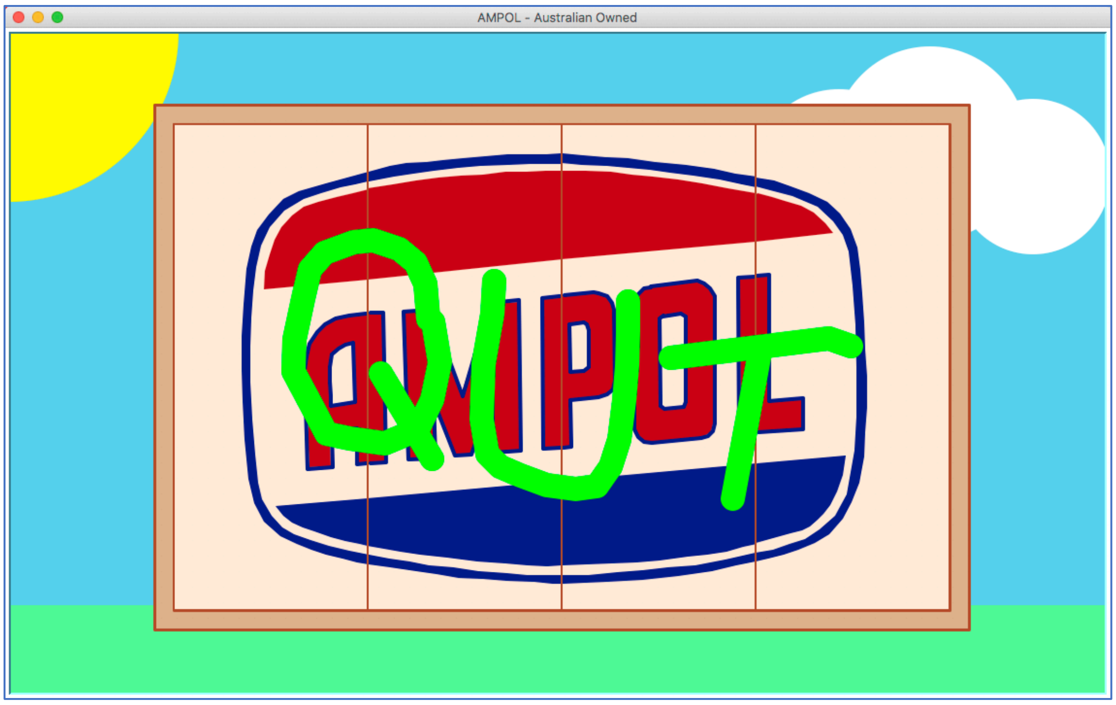
To illustrate the requirements we'll continue our example from the Part A instructions. Recall that we created four sheets which, when arranged correctly, produced the old AMPOL logo.

However, consider the following data set, which is at position 51 in our collection of pre-defined data sets.

```
['X', ['Sheet A', 'Location 1', 'Upright'],
      ['Sheet B', 'Location 2', 'Upright'],
      ['Sheet C', 'Location 3', 'Upright'],
      ['Sheet D', 'Location 4', 'Upright']]
```

[1] The singular form of graffiti is graffito, although graffiti is often used as a singular noun.

Normally these instructions would lead us to drawing the complete billboard image, but in this case the presence of an ‘X’ indicates that our precious billboard has been vandalised! Therefore, after we finish pasting up the sheets we draw a graffito across the whole billboard as follows.

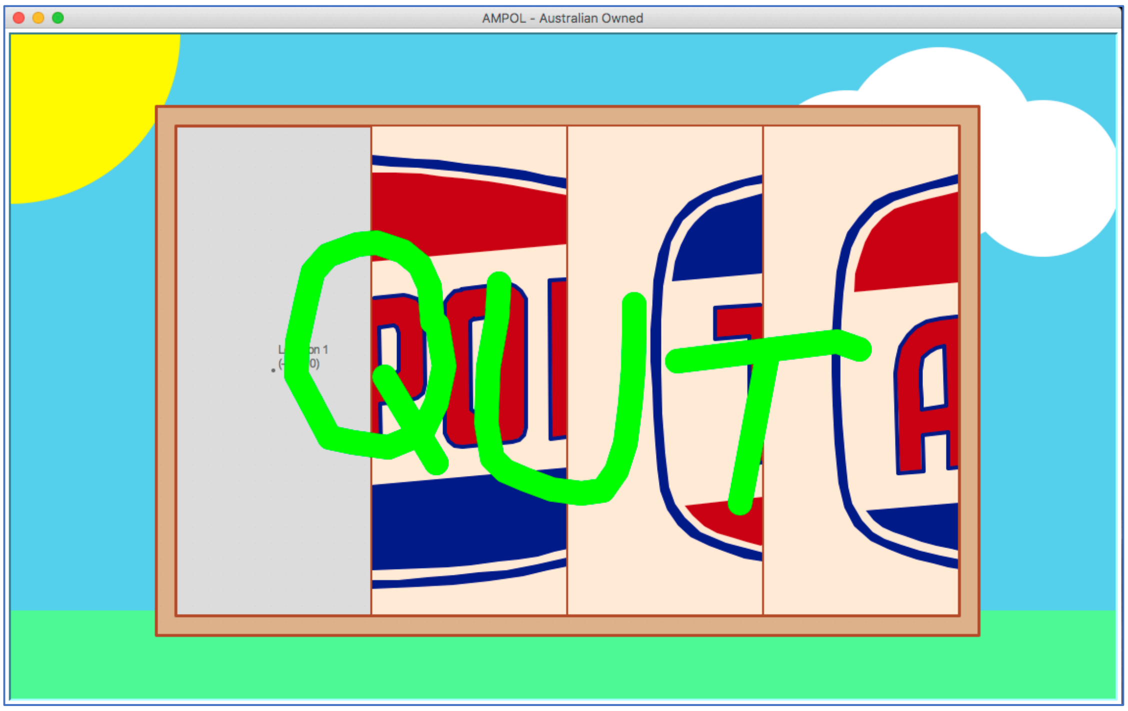


Notice that the graffito is drawn crudely, as if spray-painted, and is in a colour and style that makes it clearly distinct from the “official” billboard image. For the purposes of illustration we have used “QUT” as the initials here, but you must use **your own initials**. Zero marks will be awarded for solutions that do not use your initials as the dominate part of the graffito.

In Part A of the assignment we also allowed for the possiblity that the billposter makes mistakes, such as putting sheets in the wrong location or pasting them upside down. However, these variations make no difference to our imaginary graffiti “artist”. The graffito is always “sprayed” across the whole billboard in exactly the same way, regardless of the other instructions in the data set. For instance, consider the following data set, which is at position 43 in our collection of pre-defined data sets, and requires the resulting billboard to be graffitied.

```
['X', ['Sheet A', 'Location 4', 'Upright'],
      ['Sheet D', 'Location 3', 'Upside down'],
      ['Sheet C', 'Location 2', 'Upright']]
```

In this case we paste up only three sheets, leaving one location empty, and one of the sheets is upside down. Despite this, the graffiti tag is still sprayed across the whole billoboard in its usual orientation, resulting in the image shown overleaf.



Requirements and marking guide

To complete this task you are required to extend your Part A `billboard.py` file by modifying the code so that data sets that begin with an '`X`' result in your **personalised** graffiti tag, comprising **your initials**, being drawn across the billboard.

Your submitted solution for both Parts A and B will consist of a *single Python file*. Your Part B extension must satisfy the following criteria. Marks available are as shown.

1. **Data sets marked with an 'X' are graffitied (5%)**. When your `paste_up` function is given a data set beginning with an '`X`' it must draw your graffiti tag across the billboard. Your solution must have the following characteristics.
   - The graffito must be a "tag" consisting of **your initials, at least two**, as per your name in the statement at the beginning of the Python file. **Zero marks** will be awarded for solutions that do not clearly show your initials, regardless of any other characteristics of the solution.
   - **Only data sets beginning with an '`X`'** should be vandalised in this way.
   - The graffito must be **drawn, not printed**, as if hurriedly scrawled with a spray can. You may **not** use Turtle's `write` function to print the graffito.
   - The graffito must be large enough to cover **all four locations** on the billboard.
   - The graffito must be in colours and a style that makes it **clearly distinct from the original billboard image**.

You must complete this task using *basic Turtle graphics and maths functions only*. You may not import any additional modules or files into your program other than those already included in the original `billboard.py` template.

Development hints

- It should be possible to complete this task merely by *adding* code to your existing solution, with little or no change to the code you have already completed.
- If you are unable to complete the whole task, just submit whatever part you can get working. You will receive *partial marks for incomplete solutions*. Try to ensure that your program runs when submitted, even if it is incomplete.

Deliverable

You must develop your solution by completing and submitting the provided Python 3 file `billboard.py` as follows. **Do not submit any other files!** **Do not submit a compressed archive ('zip' or 'rar') containing multiple files!**

1. Complete the "statement" at the beginning of the Python file to confirm that this is your own individual work by inserting your name and student number in the places indicated. *We will assume that submissions without a completed statement are not your own work.*
2. Complete your solution by developing Python code to replace the dummy `paste_up` function. You must complete your solution using *only the modules already imported by the provided template*. You may *not* use or import any other modules to complete this program. In particular, you may *not* import any image files into your solution.
3. Submit *a single Python file* containing your solution for marking only. Do *not* submit an archive (e.g., in 'zip' or 'rar' formats) containing several files. Only a single file will be assessed, so you cannot accompany your solution with other files or pre-defined images.

Apart from working correctly your program code must be well-presented and easy to understand, thanks to (sparse) commenting that explains the *purpose* of significant code segments and *helpful* choices of variable and function names. *Professional presentation* of your code will be taken into account when marking this assignment.

If you are unable to solve the whole problem, submit whatever parts you can get working. You will receive *partial marks for incomplete solutions*.

How to submit your solution

A link is available on Blackboard under *Assessment* for uploading your solution file before the deadline (5:00pm Wednesday, September 13th). You can submit as many drafts of your solution as you like. You are strongly encouraged to *submit draft solutions* before the deadline as insurance against computer and network failures.