Assignment 1, Part A: Cityscapes

(20%, due 11:59pm Sunday, April 22nd, end of Week 7)

Overview

This is the first part of a two-part assignment. This part is worth 20% of your final grade for IFB104. Part B will be worth a further 5%. Part B is intended as a last-minute extension to the assignment, thereby testing the maintainability of your code, and the instructions for completing it will not be released until Week 7. Whether or not you complete Part B you will submit only one file, and receive only one assessment, for the whole 25% assignment.

Motivation

One of the most basic functions of any IT system is to process a given data set to produce some form of human-readable output. This assignment requires you to produce a visual image by following instructions stored in a list. It tests your abilities to:

- Process lists of data values;
- Design a solution to a highly-repetitive computational problem;
- Display information in a visual form; and
- Produce maintainable, reusable code.

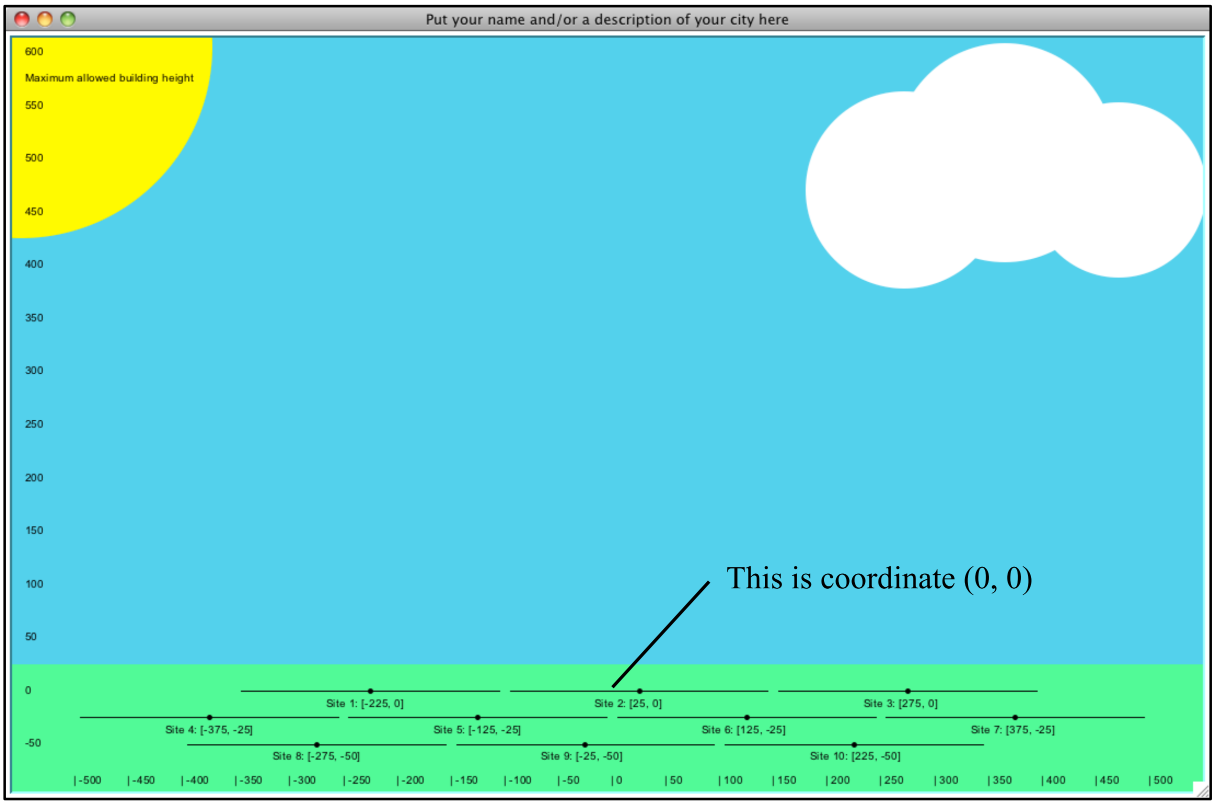
In particular, because this task is highly-repetitive it would be extremely tedious to solve it in a "brute-force" manner using lots of duplicated code. Instead you will need to think carefully about how to reuse parts of your solution, via well-planned function definitions and the use of iterative code, to make your own workload lighter and the resulting program more concise and easier to understand and maintain.

Goal

Working at QUT's inner-city Gardens Point campus gives us a good view of Brisbane's cityscape, consisting of numerous high-rise buildings in many different styles and of many different heights. In this assignment you are required to use Python's Turtle graphics module to draw such a cityscape. To do so you must follow a set of instructions, in the form of a Python list, which specify where each building is to be placed, the specific style of the building, and the number of floors. Most importantly, this "city plan" will be generated *randomly*, so your solution must be sufficiently general that it can work correctly for *any possible plan* in this format.

Resources provided

A template Python 3 program, `cityscapes.py`, is provided with these instructions. When run it creates a drawing canvas and displays a simple background image on which you will draw your buildings. The default image drawn by the provided template appears as shown overleaf. It consists of a simple image of the greenfield location where the newly planned city will be built.



This is coordinate (0, 0)

Note that in this template "home" coordinate (0, 0) is set low down in the "grassy" area, rather than the usual default of the drawing canvas's centre. Also shown are the *x* and *y* axes (measured in pixels) and ten pre-determined "building sites", from Site 1 to Site 10. The coordinates for the centre of each site are marked. This grid of coordinates is designed to help you position and size the drawings of your buildings. (To produce an uncluttered image you can disable drawing of the coordinate grid by setting the parameter to function `create_drawing_canvas` to `False` in the template's main program.)

Your task is to extend this template file so that it can draw cityscapes on this empty field. To do so you must design four entirely distinct styles of building which can be drawn on any site and at any height from one floor to ten. Your code will consist of a function called `build_city` (and any auxiliary functions you create to support it). This function takes a single argument, which is a list of instructions specifying which buildings to draw and where. This "city plan" is created by a provided function called `random_plan` which *randomly* generates the instructions, so your code must work correctly for *any possible* city plan!

Data format

The `random_plan` function which will be used to assess your solution returns "city plans" as a list of building specifications. Each specification consists of the following parts:

1. The site on which to erect the building, from Site 1 to 10.
2. The style of building to be erected on that site, from style '`A`' to '`D`'.
3. The number of floors to be constructed, from 1 to 10.

4. An extra value, either 'X' or 'O', whose purpose will be revealed only in Part B of the assignment. You should ignore it while completing Part A.

After you have run the template file you can call 'random_plan()' in IDLE's shell window to see the kinds of data it returns. Your build_city function must accept *any* such value and follow the instructions it contains to draw the corresponding cityscape. (By default function random_plan both returns and prints the city plan, but you can disable printing of the plan by supplying False as its optional argument.)

For instance, a typical list returned by function random_plan is as follows.

```
[[2, 'D', 6, 'X'],
 [3, 'C', 5, 'O'],
 [5, 'D', 8, 'O'],
 [6, 'C', 2, 'X'],
 [7, 'A', 1, 'O']]
```

This list tells us to draw: a building on Site 2, of Style D, 6 stories high; a building on Site 3 of Style C, 5 stories high; a building on Site 5, also of Style D, 8 stories high; etc.

In addition to the random_plan function, the template file also contains a number of "fixed" data sets. These are provided to help you develop your code, so that you can work with a known city plan while debugging your code. However, the "fixed" data sets will not be used for assessing your solution. Your build_city function must work for *any* city plan randomly generated by function random_plan.

Building designs

To complete this assignment you must design four *entirely distinct* styles of building, corresponding to the styles 'A' to 'D' in the city plans generated by function random_plan. You have a free choice in the visual design of your buildings and you are strongly encouraged to be imaginative! Ideally the individual buildings should fit some unified theme for the city as a whole.

Each building must have two basic features:

- A series of *stories* or *floors*, which can be varied in number from 1 to 10, as dictated by the building specification. Each story must consist of several Turtle graphics shapes representing windows, doors, ledges, fire escapes, air conditioning units or anything else you deem necessary to produce a realistic image. All the floors can be the same for a particular building style, but the floors for each building style must be *clearly distinct* from all the others, involving different Turtle graphics shapes.
- A *roof structure*, which caps the building regardless of the number of stories. Each of the four styles of building must have a distinct roof structure, and each roof structure must comprise multiple Turtle graphics shapes. Each building style must have a roof structure which is clearly different from all the others, involving different Turtle graphics shapes. Features you could incorporate into the roof structures include domes, spires, aerials, satellite dishes, advertising signs, chimneys, exhaust pipes, helipads, revolving restaurants, etc.

You also have a free choice of the dimensions for the individual building styles. The best visual effect for the overall cityscape will be achieved if each style has different dimensions.

Also, we assume the city council has imposed some constraints on the maximum size of the buildings:

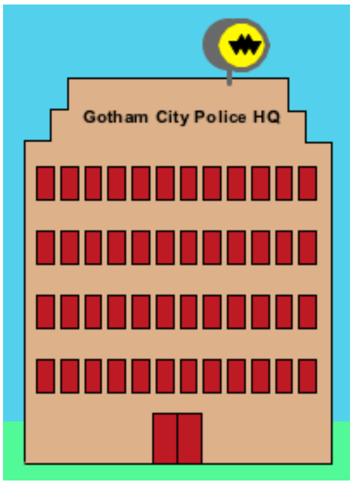
- Buildings cannot be wider than 250 pixels (which is the maximum space available at each building site).
- Each story of each building should not be more than about 50 pixels high (because a building of 10 stories, excluding the roof structure, would then be 500 pixels high, which is close to the maximum legal building height shown in the grid).

At the other extreme, of course, each building must be wide and high enough to make all of its individual features easily visible.

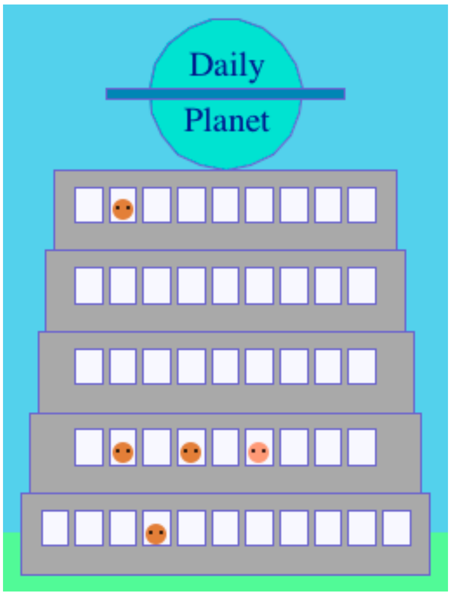
Illustrative example

To illustrate the requirements we developed a solution which uses famous buildings from science-fiction films and comics as its theme. (You should *not* copy our example! Develop your own idea! Be imaginative!) We developed code to draw the four following distinct styles of building.

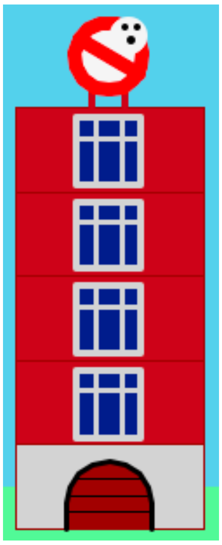
Style A:



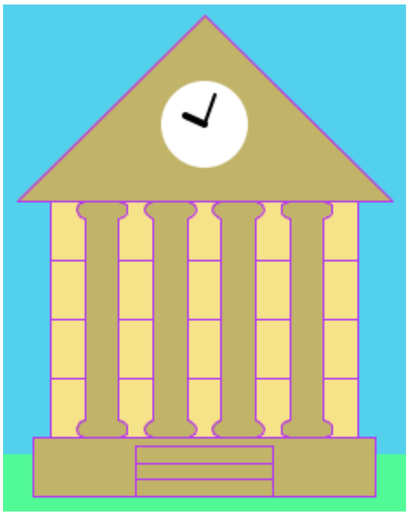
Style B:



Style C:



Style D:



Each of the building styles has been shown at a height of 5 stories above, but our solution allows each of them to be drawn at any height from 1 to 10 stories, inclusive. Each building has clearly distinct floors and roof structures and distinct vertical and horizontal dimensions. (The four images above are all reproduced at the same scale.)

The four building styles we created are as follows:

A. The Gotham City Police headquarters from *Batman*. Each floor has multiple windows, except the ground floor which has doors. The roof contains signage and the Batsignal, ready to summon the Caped Crusader and the Boy Wonder to action.

B. The Daily Planet building from *Superman*. Here the floors are all the same but taper in width as we go up. Reporters are shown looking anxiously out the windows for Metropolis' hero. (The people are placed in the windows randomly and have random skin tones.) The roof contains the famous Daily Planet logo consisting of a ringed planet.

C. The converted firehouse from *Ghostbusters* (which is a real building in New York). Again the ground floor has been made different from the others, to show the roller door for the Ectomobile. The roof contains the Ghostbusters' logo (although in the movie this sign actually hangs above the door).

D. The clock tower from *Back to the Future*. Here we have used columns that span all the stories as the distinctive shape of each floor, except the ground floor which is a plinth containing stairs. The roof structure contains the clock, which is stopped at precisely 10:04pm, the time it was struck by lightning in 1955.

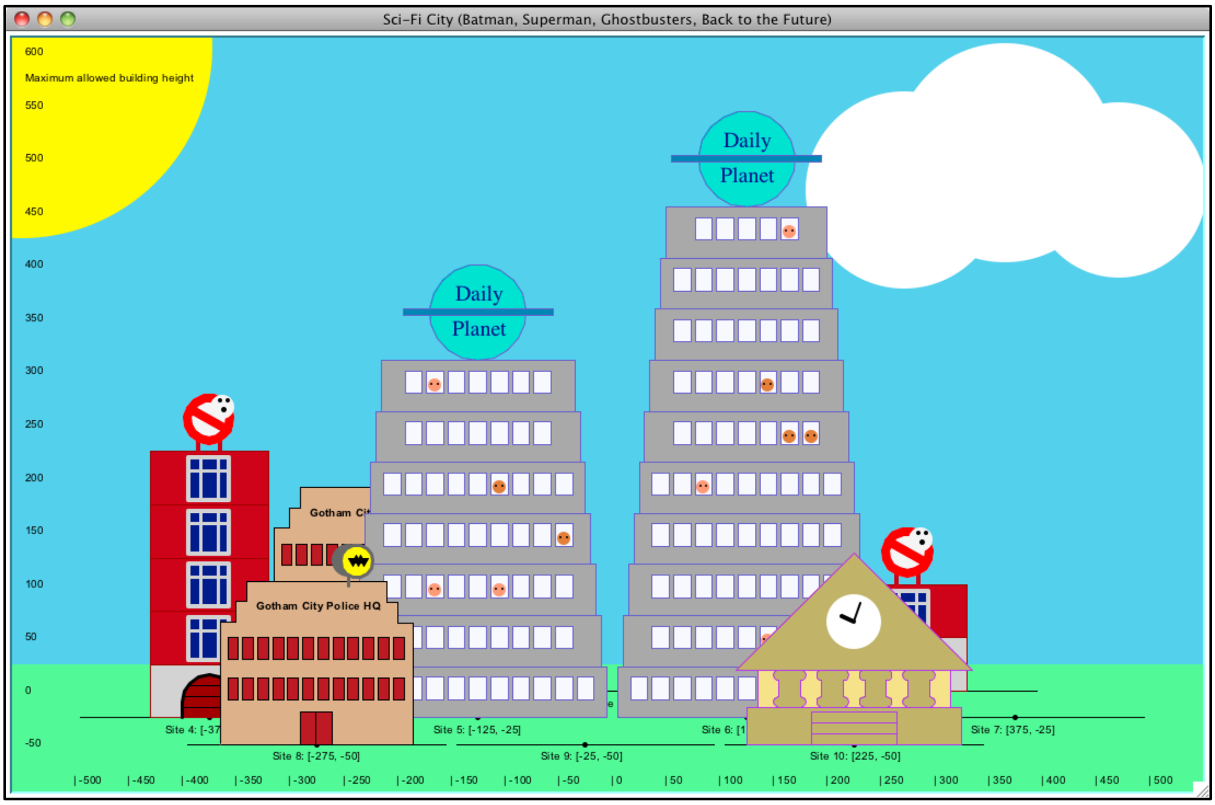
Note that our demonstration solution is more elaborate than necessary. Random elements (the people looking out of the Daily Planet's windows) are not necessary, and all the floors can be the same in each style of building, including the ground floor (although this does not produce the most realistic looking image), provided they are reasonably complex.

From this basis, our implementation of the `build_city` function can draw any city plan generated by function `random_plan`. For instance, consider the following city plan:

```
[[1, 'A', 4, 'O'],
 [3, 'C', 2, 'O'],
 [4, 'C', 5, 'O'],
 [5, 'B', 7, 'O'],
 [6, 'B', 10, 'O'],
 [8, 'A', 3, 'O'],
 [10, 'D', 2, 'O']]
```

This requires us to erect buildings of style A (the Gotham City Police HQ) on sites 1 and 8, of heights 4 stories and 3 stories, respectively. We are also required to put a copy of building style D (the clock tower), only two floors high, on site 10. Buildings of style B (the Daily Planet) of different heights are specified for sites 5 and 6, and so on.

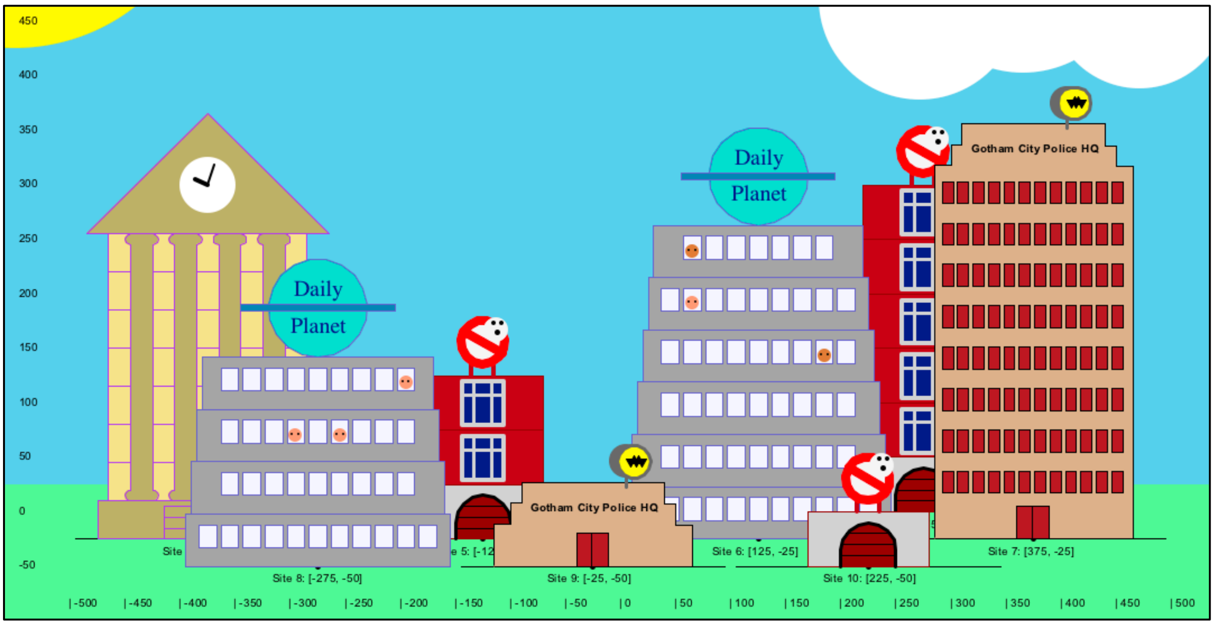
The resulting image is shown overleaf. Notice in each case that a building of the appropriate style and number of floors is drawn precisely centred on the specified building site.



As another example, consider the following data set:

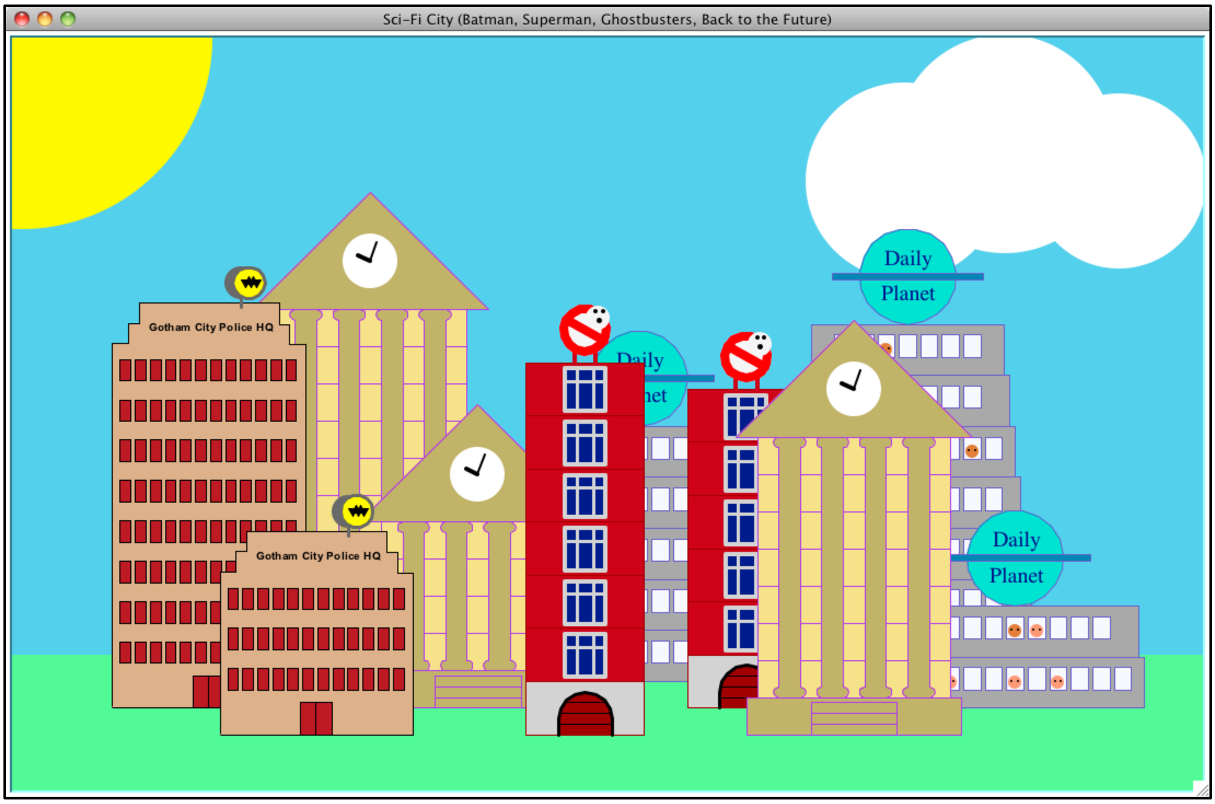
```
[[3, 'C', 6, 'O'], [4, 'D', 8, 'O'], [5, 'C', 3, 'O'],
 [6, 'B', 6, 'O'], [7, 'A', 9, 'O'], [8, 'B', 4, 'O'],
 [9, 'A', 1, 'O'], [10, 'C', 1, 'O']].
```

The cityscape drawn in this case is as follows, again following precisely the building specifications.



Once we are fully satisfied with our solution, we can change the argument to the `create_drawing_canvas` function to omit the grid, to produce a more attractive picture. The cityscape below was created for the following dataset, with the grid switched off.

```
[[1, 'D', 10, 'O'], [2, 'B', 5, 'O'], [3, 'B', 7, 'O'],
 [4, 'A', 9, 'O'], [5, 'D', 5, 'O'], [6, 'C', 6, 'O'],
 [7, 'B', 2, 'O'], [8, 'A', 4, 'O'], [9, 'C', 7, 'O'],
 [10, 'D', 8, 'O']]
```



Requirements and marking guide

To complete this part of the assignment you are required to extend the provided `cityscapes.py` Python file by completing function `build_city` so that it can draw buildings at the sites and heights specified by the data sets generated by the `random_plan` function. Your `build_city` function must work correctly for any values returned by the `random_plan` function.

Your submitted solution will consist of a single Python 3 file, and must satisfy the following criteria. Percentage marks available are as shown.

1. **Drawing four entirely distinct building styles (5%).** Your program must be able to draw four distinct styles of building, each consisting of multiple floors and a roof structure, such that each building is clearly distinct in all elements from the other styles of building. Each floor and roof structure must be of a reasonable degree of complexity, involving multiple Turtle shapes. It must be easy to distinguish each floor from those above and/or below it, so that the number of floors in any particular building can be counted. Similarly, the roof structures must be clearly distinct from

the floors below. Also, the buildings must be made sufficiently distinct, either by drawing borders and/or varying their colours, so that they can be distinguished from one another even when they overlap on the canvas.

Beyond these requirements you are free to add other variations to your building designs, such as varying the width of the floors, making the ground floor different from the others, adding different features for different floors, etc.

2. **Positioning buildings (5%)**. Your code must be capable of drawing each of the four styles of building at any of the marked building sites, as dictated by the data set provided to function `build_city`. The buildings must be centred precisely on the specified coordinates. The drawings of the buildings must preserve their integrity no matter where they are drawn, with no spurious additional or missing lines. Your solution for relocating the buildings must work correctly for *any* values returned by the `random_plan` function.

3. **Resizing buildings (5%)**. Your code must be capable of drawing each of the four building styles at any height, as specified by the number of floors in the data set provided to function `build_city`. The correct number of floors must be drawn and always be topped by the appropriate roof structure. The individual parts of the building must preserve their integrity no matter how tall or short it is, with no spurious additional or missing lines. Your solution for resizing the buildings must work correctly for *any* values returned by the `random_plan` function.

4. **Code quality and presentation (5%)**. Your program code, for both Parts A and B of the assignment, must be presented in a professional manner. See the coding guidelines in the *IFB104 Code Presentation Guide* (on Blackboard under *Assessment*) for suggestions on how to achieve this. In particular, given the obscure and repetitive nature of the code needed to draw complex images using Turtle graphics, each significant code segment must be clearly commented to say what it does, e.g., "Draw door", "Draw ghost's head", etc. Similarly, the names of your functions and variables should be indicative of their purpose, not just "`i`", "`j`", etc. Also, you must use function definitions and loops to avoid unnecessary duplication of similar or identical code segments.

5. ***Extra feature (5%)**. Part B of this assignment will require you to make a last-minute extension to your solution. The instructions for Part B will not be released until shortly before the final deadline for Assignment 1.*

You must complete the assignment 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 given `cityscapes.py` template. In particular, you may not import any image files for use in creating your drawing.

Finally, you are *not* required to copy the example shown in this document. Instead you are strongly encouraged to be creative in the design of your buildings. Surprise us!

Artistic merit – The Hall of Fame!

You will not be assessed on the artistic merit of your solution, only the ability to create four entirely distinct buildings. However, a "Hall of Fame" containing the solutions considered the most artistic or ambitious by the assignment markers will be created on Blackboard. (Sadly, additional marks will not be awarded to the winners, only kudos.)

Development hints

- This is not a difficult assignment, but due to the need to create multiple building styles and multiple floors for each building it would be very tedious if you tried to code the whole solution using 'brute force'. Instead you are strongly encouraged to define parameterised functions to draw various common elements of the buildings, and use loops to call these functions, to reduce the overall amount of code you need to write.
- If you are unable to complete the whole task, just submit whatever you can get working. You will receive *partial marks* for incomplete solutions.
- To help you debug your code for the individual building styles we have provided some "fixed" datasets. Feel free to use these when developing your program, and add additional ones if you like, but keep in mind that these datasets will not be used for assessing your solution. Your `build_city` function must work for *any* value that can be returned by function `random_plan`.
- Part B of this assignment will require you to change your solution slightly in a short space of time. You are therefore encouraged to keep code maintainability in mind while developing your solution to Part A. Make sure your code is neat and well-commented so that you will find it easy to modify when the instructions for Part B are released.

Deliverable

You must develop your solution by completing and submitting the provided Python 3 file `cityscapes.py` as follows.

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 `build_city` function. You should complete your solution using only the standard Python 3 modules already imported by the provided template. In particular, you must *not* use any Python modules that must be downloaded and installed separately because the markers may not have these modules installed. Furthermore, you may *not* import any image files into your solution; the entire image must be drawn using Turtle graphics drawing primitives.
3. Submit *a single Python file* containing your solution for marking. Do *not* submit multiple files. Only a single file will be accepted, so you cannot accompany your solution with other files or pre-defined images. **Do not submit any other files!** **Submit only a single Python 3 file!**

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*.

Plagiarism

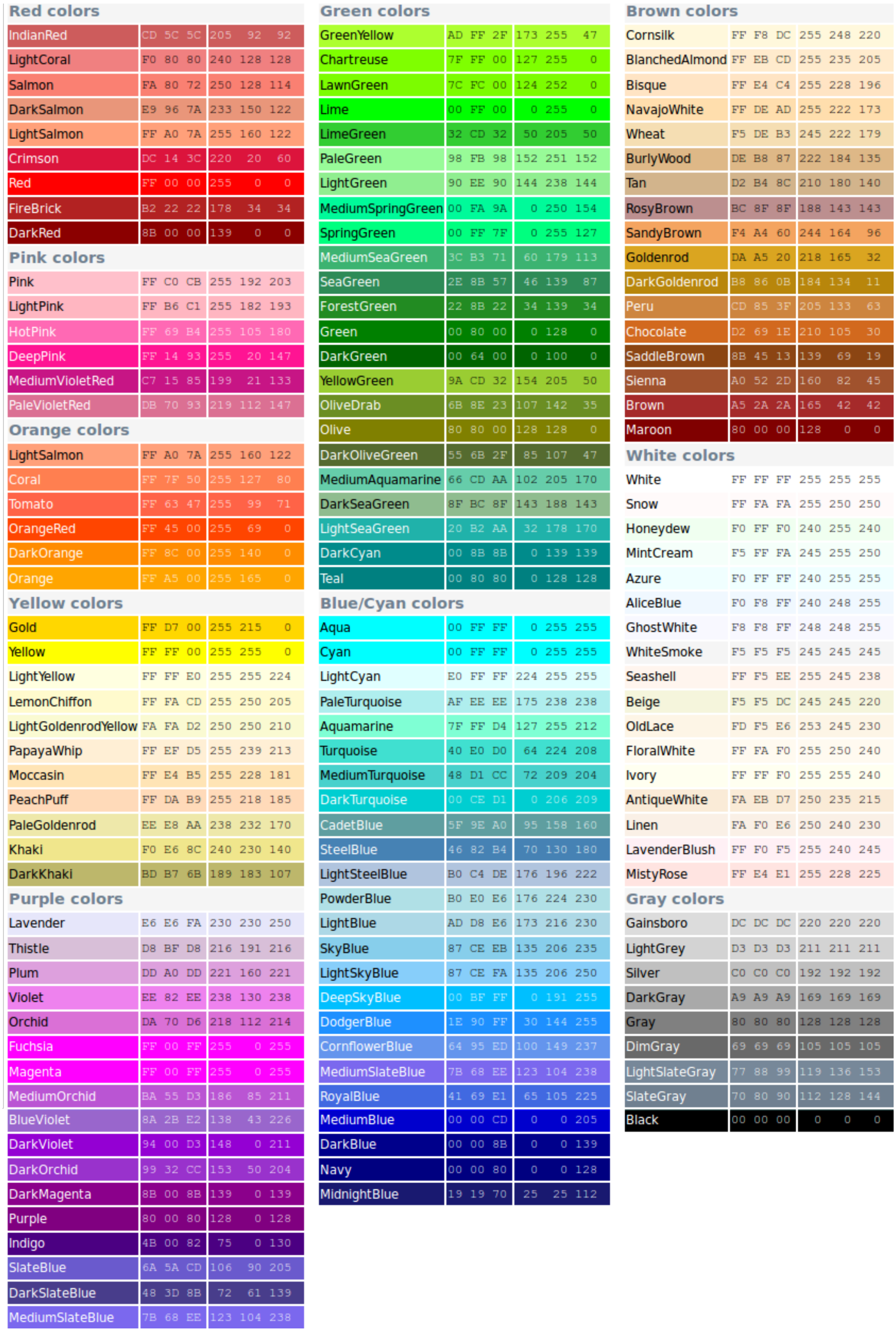
This is an individual assessment item. All files submitted will be subjected to software plagiarism analysis using the MoSS system (http://theory.stanford.edu/~aiken/moss/). Serious violations of the university's policies regarding plagiarism will be forwarded to the Science and Engineering Faculty's Academic Misconduct Committee for formal prosecution.

How to submit your solution

A link will be made available on the IFB104 Blackboard site under *Assessment* for uploading your solution file before the deadline (11:59pm Sunday, April 22nd, end of Week 7). 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 or network problems near the deadline. If you are unsure whether or not you have successfully uploaded your file, upload it again!

Students who encounter problems uploading their Python files to Blackboard should contact the *IT Helpdesk* (ithelpdesk@qut.edu.au; 3138 4000) for assistance and advice. Teaching staff will *not* answer email queries on the weekend the assignment is due, so ensure that you have successfully uploaded at least one solution by close-of-business on Friday, April 20th.

Appendix: Some standard Turtle graphics colours you can use



Red colors

IndianRed	CD 5C 5C	205 92 92
LightCoral	F0 80 80	240 128 128
Salmon	FA 80 72	250 128 114
DarkSalmon	E9 96 7A	233 150 122
LightSalmon	FF A0 7A	255 160 122
Crimson	DC 14 3C	220 20 60
Red	FF 00 00	255 0 0
FireBrick	B2 22 22	178 34 34
DarkRed	8B 00 00	139 0 0

Pink colors

Pink	FF C0 CB	255 192 203
LightPink	FF B6 C1	255 182 193
HotPink	FF 69 B4	255 105 180
DeepPink	FF 14 93	255 20 147
MediumVioletRed	C7 15 85	199 21 133
PaleVioletRed	DB 70 93	219 112 147

Orange colors

LightSalmon	FF A0 7A	255 160 122
Coral	FF 7F 50	255 127 80
Tomato	FF 63 47	255 99 71
OrangeRed	FF 45 00	255 69 0
DarkOrange	FF 8C 00	255 140 0
Orange	FF A5 00	255 165 0

Yellow colors

Gold	FF D7 00	255 215 0
Yellow	FF FF 00	255 255 0
LightYellow	FF FF E0	255 255 224
LemonChiffon	FF FA CD	255 250 205
LightGoldenrodYellow	FA FA D2	250 250 210
PapayaWhip	FF EF D5	255 239 213
Moccasin	FF E4 B5	255 228 181
PeachPuff	FF DA B9	255 218 185
PaleGoldenrod	EE E8 AA	238 232 170
Khaki	F0 E6 8C	240 230 140
DarkKhaki	BD B7 6B	189 183 107

Purple colors

Lavender	E6 E6 FA	230 230 250
Thistle	D8 BF D8	216 191 216
Plum	DD A0 DD	221 160 221
Violet	EE 82 EE	238 130 238
Orchid	DA 70 D6	218 112 214
Fuchsia	FF 00 FF	255 0 255
Magenta	FF 00 FF	255 0 255
MediumOrchid	BA 55 D3	186 85 211
BlueViolet	8A 2B E2	138 43 226
DarkViolet	94 00 D3	148 0 211
DarkOrchid	99 32 CC	153 50 204
DarkMagenta	8B 00 8B	139 0 139
Purple	80 00 80	128 0 128
Indigo	4B 00 82	75 0 130
SlateBlue	6A 5A CD	106 90 205
DarkSlateBlue	48 3D 8B	72 61 139
MediumSlateBlue	7B 68 EE	123 104 238

Green colors

GreenYellow	AD FF 2F	173 255 47
Chartreuse	7F FF 00	127 255 0
LawnGreen	7C FC 00	124 252 0
Lime	00 FF 00	0 255 0
LimeGreen	32 CD 32	50 205 50
PaleGreen	98 FB 98	152 251 152
LightGreen	90 EE 90	144 238 144
MediumSpringGreen	00 FA 9A	0 250 154
SpringGreen	00 FF 7F	0 255 127
MediumSeaGreen	3C B3 71	60 179 113
SeaGreen	2E 8B 57	46 139 87
ForestGreen	22 8B 22	34 139 34
Green	00 80 00	0 128 0
DarkGreen	00 64 00	0 100 0
YellowGreen	9A CD 32	154 205 50
OliveDrab	6B 8E 23	107 142 35
Olive	80 80 00	128 128 0
DarkOliveGreen	55 6B 2F	85 107 47
MediumAquamarine	66 CD AA	102 205 170
DarkSeaGreen	8F BC 8F	143 188 143
LightSeaGreen	20 B2 AA	32 178 170
DarkCyan	00 8B 8B	0 139 139
Teal	00 80 80	0 128 128

Blue/Cyan colors

Aqua	00 FF FF	0 255 255
Cyan	00 FF FF	0 255 255
LightCyan	E0 FF FF	224 255 255
PaleTurquoise	AF EE EE	175 238 238
Aquamarine	7F FF D4	127 255 212
Turquoise	40 E0 D0	64 224 208
MediumTurquoise	48 D1 CC	72 209 204
DarkTurquoise	00 CE D1	0 206 209
CadetBlue	5F 9E A0	95 158 160
SteelBlue	46 82 B4	70 130 180
LightSteelBlue	B0 C4 DE	176 196 222
PowderBlue	B0 E0 E6	176 224 230
LightBlue	AD D8 E6	173 216 230
SkyBlue	87 CE EB	135 206 235
LightSkyBlue	87 CE FA	135 206 250
DeepSkyBlue	00 BF FF	0 191 255
DodgerBlue	1E 90 FF	30 144 255
CornflowerBlue	64 95 ED	100 149 237
MediumSlateBlue	7B 68 EE	123 104 238
RoyalBlue	41 69 E1	65 105 225
MediumBlue	00 00 CD	0 0 205
DarkBlue	00 00 8B	0 0 139
Navy	00 00 80	0 0 128
MidnightBlue	19 19 70	25 25 112

Brown colors

Cornsilk	FF F8 DC	255 248 220
BlanchedAlmond	FF EB CD	255 235 205
Bisque	FF E4 C4	255 228 196
NavajoWhite	FF DE AD	255 222 173
Wheat	F5 DE B3	245 222 179
BurlyWood	DE B8 87	222 184 135
Tan	D2 B4 8C	210 180 140
RosyBrown	BC 8F 8F	188 143 143
SandyBrown	F4 A4 60	244 164 96
Goldenrod	DA A5 20	218 165 32
DarkGoldenrod	B8 86 0B	184 134 11
Peru	CD 85 3F	205 133 63
Chocolate	D2 69 1E	210 105 30
SaddleBrown	8B 45 13	139 69 19
Sienna	A0 52 2D	160 82 45
Brown	A5 2A 2A	165 42 42
Maroon	80 00 00	128 0 0

White colors

White	FF FF FF	255 255 255
Snow	FF FA FA	255 250 250
Honeydew	F0 FF F0	240 255 240
MintCream	F5 FF FA	245 255 250
Azure	F0 FF FF	240 255 255
AliceBlue	F0 F8 FF	240 248 255
GhostWhite	F8 F8 FF	248 248 255
WhiteSmoke	F5 F5 F5	245 245 245
Seashell	FF F5 EE	255 245 238
Beige	F5 F5 DC	245 245 220
OldLace	FD F5 E6	253 245 230
FloralWhite	FF FA F0	255 250 240
Ivory	FF FF F0	255 255 240
AntiqueWhite	FA EB D7	250 235 215
Linen	FA F0 E6	250 240 230
LavenderBlush	FF F0 F5	255 240 245
MistyRose	FF E4 E1	255 228 225

Gray colors

Gainsboro	DC DC DC	220 220 220
LightGrey	D3 D3 D3	211 211 211
Silver	C0 C0 C0	192 192 192
DarkGray	A9 A9 A9	169 169 169
Gray	80 80 80	128 128 128
DimGray	69 69 69	105 105 105
LightSlateGray	77 88 99	119 136 153
SlateGray	70 80 90	112 128 144
Black	00 00 00	0 0 0