Duration: **50 minutes**
Aids Allowed: ***None***

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This term test consists of 5 questions on 10 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

In your answers, you may use any of the Python builtin functions and standard modules listed on the accompanying Python reference sheet. You must write your own code for everything else.

Comments and docstrings are *not required*, except where indicated, although they may help us grade your answers. Also, they may be worth part marks if you cannot write a complete answer.

If you are unable to answer a question (or part of a question), remember that you will get 10% of the marks for any solution that you leave *entirely blank* (or where you cross off everything you wrote to make it clear that it should not be marked).

MARKING GUIDE

1: _____/ 6

2: _____/ 9

3: _____/ 8

4: _____/ 7

5: _____/10

TOTAL: _____/40

Good Luck!

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 1. [6 MARKS]

Write the body of the function below to satisfy its docstring. Assume that module `queue.py` defines a class `Queue` that provides the usual methods: `is_empty()`, `enqueue(item)`, `dequeue()`, `front()`.

Your code must *not* depend on any detail of the implementation of class `Queue`. In other words, the only thing you can do with `Queue` objects is to call some of their methods.

```
from queue import Queue

def size(que):
    """(Queue) -> int
    Return the number of items on Queue que. The contents of que are unchanged
    (it's OK if the contents of que are modified during the execution of this
    function, as long as everything is restored before the function returns).
    """
    # Hint: Create a second Queue object to use for temporary storage.
```

Question 2. [9 MARKS]

Write classes that satisfy the following specification. Docstrings are *not* required.

- A `Car` has a `model` (an arbitrary string) and a `year` or production (an integer in the range 1770–2013).
- Attempting to create a `Car` object with a year of production outside of the prescribed range `raises` an `InvalidYearError`.
- `Car` objects have only one behaviour: the string representation of a `Car` consists of the `Car`'s `model` followed by a space followed by the `Car`'s `year` of production.

Question 3. [8 MARKS]

Write the body of the function below to satisfy its docstring. Assume this function is written in the same file as your code from the previous question (so nothing needs to be `import`ed).

```
def create_car():
    """() -> Car
    Ask the user for a model and a year, using the input() function, then return
    a new Car object with that model and year of production. If the user enters
    an invalid year, ask again (repeatedly) until a valid year is entered.
    """
    # For full marks, don't test the validity of the year directly -- just try
    # to create a Car and handle the possible outcomes appropriately.
```

Question 4. [7 MARKS]

Write the body of the function below to satisfy its docstring. (HINT: Recursion makes this simple!)

```
def nest_level(L):
    """(nested list) -> int
    Return the maximum "nesting level" of nested list L, that is, how many
    levels of sub-lists are in L, at the "deepest" point within L.
    For example, nest_level([1, [], [[2]]], 'three']) == 4 because element 2
    is in a list ([2]) within a list ([[2]]) within a list ([[], [[2]]]) within
    the original list.
    """
```

Question 5. [10 MARKS]

Write code on the next page for a complete `unittest` file for function `nest_level` above. *Include a brief (one line) docstring for each test function*, to explain the purpose of the test. For full marks, you should have at least *three* test cases involving some nested and some non-nested inputs.

Question 5. (CONTINUED)

```
import unittest
from q2 import nest_level  # assume nest_level is in a file named "q2.py"
```

```
if __name__ == '__main__':
    unittest.main(exit=False)
```

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

On this page, please write nothing except your name.

Last (Family) Name(s): _____

First (Given) Name(s): _____

Total Marks = 40