3 A programmer is writing a program to manipulate different data structures using Object-Oriented Programming (OOP).

The superclass, `DataStructure`, will store the following data:

- a linked list of the data items
- head pointer, pointing to the first element in the linked list
- tail pointer, pointing to the last element in the linked list

This class has one method to display all the current contents in the structure, in the order they are stored in the linked list.

The superclass is used to implement a stack and a linear queue.

The subclass `Stack` has the following methods:

- `insert(value)` appends the parameter to the stack.
- `delete()` returns and removes the next value in the stack.
- `display()` method should display the stack in reverse order (e.g. the most recently added element first) and should override the `DataStructure` display method.

The subclass `Queue` has the following methods:

- `insert(value)` appends the parameter value to the queue.
- `delete()` returns and removes the next value in the queue.
- `display()` method should display the queue contents in order (e.g. the earliest added element first) and should override the `DataStructure` display method.

Each method updates its appropriate pointers, and produces suitable errors (or returns different values) to indicate if the actions are not possible, e.g. if the structure is empty.

For each of the sub-tasks, add a comment statement, at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

```
In [1]: #Task 3.1
        Program code
```

```
In [2]: #Task 3.2
        Program code
```

```
In [3]: #Task 3.3
        Program code
```

```
In [4]: #Task 3.4
        Program code
```

```
Output:
```

Task 3.1

Write program code for the superclass `DataStructure`. [3]

Task 3.2

Write program code for the subclass `Stack`.

Use appropriate inheritance and polymorphism in your designs. [5]

Task 3.3

Write program code for the subclass `Queue`.

Use appropriate inheritance and polymorphism in your designs. [5]

Task 3.4

The files `TASK3stack.txt` and `TASK3queue.txt` store data to test your program.

Write program code to:

- create a new stack and add the data in the file `TASK3stack.txt` to the stack
- create a new queue and add the data in the file `TASK3queue.txt` to the queue
- output the current contents of both the stack and queue
- remove and output two items from both the stack and queue
- output the contents of both the stack and queue after the removal of the items.

All outputs should have appropriate messages to indicate what they are showing.

You are required to present the output of stack and queue both before and after the removal of items.

Download your program code and output for Task 3 as
`TASK3_<your name>_<centre number>_<index number>.ipynb` [9]