Question 1

Create a function called **display_as_list**

display_as_list has **three parameters**

- **display_items**
- **message**
- **counter_reqd**

display_items is **mandatory** and both message and counter_reqd are **optional**.

display_items is the list of items that needs to be displayed.

message is what might be displayed before each item is displayed. message is a string and **defaults** to 'Item'.
The message string must not be modified.

counter_reqd is **boolean** and **defaults** to True.

If there are items in the list, the function displays them as required and **returns True**. See the examples below.
If there are no items in the list, the function informs the user accordingly and **returns False**. See the example below.

If counter_reqd is True then on each line, display both the message and the counter. If counter_reqd is False then on each line, only display the item, not the message or the counter.

display_as_list prints a blank line before printing the individual elements of the list on separate lines as per the examples below.

Hints.

- Use print() to generate the blank line.
- The solution is between 9 and 13 lines of code ignoring blank lines, comments and the docstring.
- Ensure you test for the expected return type and value.
- Not all the test data used is shown.

Question 2

Create a function called **get_option**

get_option has one parameter.

- prompt_msg

prompt_msg is optional and defaults to 'Option: '

get_option asks the user to enter an option, removes any extra spaces, converts the entered text into lower case and **returns** it as a **string**.

Hints.

- Ignoring the docstring and comments, the solution is about 2 or 3 lines of code.
- Your code must pass all the tests in order to gain the marks available for this question.
- Not all the tests used are shown.

Question 3

Create a function called **get_item**

get_item has one parameter.

- prompt_msg

prompt_msg is optional and defaults to 'Item: '

get_item asks the user to enter an item, removes any extra spaces, capitalises the entered text and **returns** it as a **string**.

Hints.

- This requires only a slight modification to the code in the previous question.
- Ignoring the docstring and comments, the solution is about 2 or 3 lines of code.
- Your code must pass all the tests in order to gain the marks available for this question.
- Not all the tests used are shown.

Question 4

Create a function called **get_total_items**

get_total_items has one parameter:

- user_list

user_list is a list.

get_total_items returns a string which is a message showing the total number of elements there are in the list. The **string returned**is grammatically correct for the number of items. See the examples below.

Hints.

- Ignoring the docstring and comments, the solution can be between 2 and 6 lines of code.
- get_total_items returns a string, please note there is no mention of display or print in the specification above.
  get_total_items does not print anything.
- Your code must pass all the tests in order to gain the marks available for this question.
- Not all the tests used are shown.

Question 5

Create a function called **add_item**

add_item has one parameter:

- user_list

user_list is a list.

The **add_item** function makes use of
the **get_item**, **get_total_items** and **get_option** functions by **calling** or **invoking** them.

add_item asks the user to enter the item to be added by **calling get_item**.

If no item was entered, add_item informs the user of this, displays the number of items in the list by **calling get_total_items** and **returns** boolean **False**.

If the item to be added is in the list, add_item asks the user to confirm that they want to add another instance of that item by **calling get_option**.

If the user confirms they want to add another instance of the item, the item is to be added to the list and the user is informed of this, add_item displays the number of items in the list by **calling get_total_items** and **returns** boolean **True**.

If the user does not acknowledge that the item is to be added to the list, inform the user the item was not added and display the total number of items in the list
and **returns** boolean **False**.

Only 'y' or 'n' are expected as replies when prompted.

If the item to be added is not in the list, it is added to the list, the user is informed of this, add_item displays the number of items in the list
by **calling get_total_items** and **returns** boolean **True**.

If an item was added to the list, add_item displays the number of items in the list by **calling get_total_items** and **returns** boolean **True** otherwise it displays the number of items in the list by **calling get_total_items** and returns boolean **False**.

If an item was added to the list, add_item returns True.
If an item was not added to the list, add_item returns False.

Paste the code for **add_item below**. Do **not** include the code
for **get_item**, **get_total_items** or **get_option.**

Hints.

- Ignoring the docstring and comments, the solution is about 18 lines of code. It will be a little longer if you duplicate sections of code.
  Please consider removing any duplicate sections of code.
- add_item does not call the input function. add_item calls the get_item and get_option function.
- Remember what get_option returns.
- Not all the tests used are shown.

Question 6

Create a function called **remove_item**

remove_item has one parameter:

- user_list

user_list is a list.

If the list contains items, then remove_item asks the user to enter the item number to be removed by calling **get_option**. The entered item number needs to be validated. A valid item number is an integer. If invalid data was received from get_option, call get_option again and keep doing so until the user enters a valid value.

If the item number is valid and it represents an item in the list, then remove_item confirms the request by calling get_option and if the user indicates they want the item removed, remove_item deletes the associated item, informs the user it has been deleted and **returns** boolean **True**. If the user says they don't want to item removed, remove_item informs the user accordingly and **returns** boolean **False**.

If the item number is 0, remove_item informs the user the request is cancelled and **returns** boolean **False**.

If the item number is valid but does not represent an item in the list, remove_item informs the user and **returns** boolean **False**.

If the list is empty, remove_item informs the user and **returns** boolean **False**. There is no point in asking the user which item they would like to have removed if the list is empty.

Paste the code for **remove_item below**. Do **not** include the code for **get_option.**

Hints.

- Notice how entering an invalid number results in the user being asked to enter it again. Use a loop to validate the item number.
- Ignoring the docstring and comments, the solution is about 21 lines of code.
- remove_item does not call the input function. remove_item calls the get_option function.
- The user is only expected to enter an item (element) number when asked and so + or - symbols are not valid.
- Remember the difference between an item (element) number and an index number.

Question 7

Create a function called **sort_list**

sort_list has one parameter:

- user_list

user_list is a list.

If there are enough items in the list then sort_list sorts the items in the list, informs the user and **returns** boolean **True**.

If there is only one item in the list, then there's no point in sorting the list
and sort_list informs the user and **returns** boolean **False**.

If there are no items in the list then sort_list informs the user
and **returns** boolean **False**.

Hints.

- Ignoring the docstring and comments, the solution is about 10 lines of code.
- Your code must pass all the tests in order to gain the marks available for this question.

Question 8

Create a function called **empty_list**

empty_list has one parameter:

- user_list

user_list is a list.

If the list is not empty and the user confirms they want to empty the list (by entering 'y' in either case) then empty_list clears the items from the list and **returns** boolean **True**.

If the list is not empty and the user does not confirm they want to empty the list then empty_list informs the user accordingly and **returns** boolean **False**.

The empty_list function must **call** the **get_option** function to ascertain if the user wants to empty the list.

If there are no items in the list then empty_list informs the user
and **returns** boolean **False**.

Paste the code for **empty_list** below. Do **not** include the code for get_option.

Hints.

- Ignoring the docstring and comments, the solution is about 11 or 12 lines of code.
- empty_list does not call the input function. empty_list calls the get_option functions.
- Your code must pass all the tests in order to gain the marks available for this question.
- Not all the tests used are shown.

Question 9

Create a function called **count_instances**

count_instances has one parameter:

- user_list

user_list is a list.

If the list is not empty then count_instances asks the user to enter the item to be counted by calling the **get_item** function and informs the user how many instances of that item were found in the list and **returns** boolean **True**.

If the list is empty then count_instances informs the user accordingly
and **returns** boolean **False**.

Hints.

- Ignoring the docstring and comments, the solution is between 8 and 13 lines of code.
- count_instances does not call the input function. count_instances calls the get_item function.
- Not all the tests used are shown.
- Your code must pass all the tests in order to gain the marks available for this question.