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**.

message is a string and **defaults** to 'Item'.

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

Your solution needs to pass all the tests in order to score the marks for this question.

Hints.

- The function is identical to question 1 in the lists lab. Re-using that function is the preferred solution. It is a good reminder of why we write code as functions.
- 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**.

Your solution needs to pass all the tests in order to score the mark for this question.

Hints.

- The function is identical to the shopping lists lab get_option function. Re-using that function is the preferred solution. It is a good reminder of why we write code as functions.
- Not all the test code used is shown.

Question 3

Create a function called **get_name**

get_name has **one parameter**.

- **message**

message is the message to be **displayed** as a prompt for the user.

Any leading and trailing spaces need to be removed from the entered name. The entered name needs to be correctly titled.

get_name **returns** the entered name as a **string**.

Your solution needs to pass all the tests in order to score the mark for this question.

Hint.

- Ignoring the docstring, the solution can be 2 or 3 lines of code.

Question 4

Create a function called **get_number_as_string**

get_number_as_string has two parameters

- **message**
- **min_length**

message is the message to display when asking the user for input.

min_length is the minimum number of digits needed.

get_number_as_string asks the user to a number using the message it received as a parameter. All spaces must be removed from the entered value.

The number is valid if it contains only digits and is at least min_length in length.

If the value entered is comprised of not only digits, inform the user and ask for the value to be entered again. See examples below.

If the value entered is comprised of only digits but is less than the minimum required length, inform the user and ask for the value to be entered again. See examples below.

get_number_as_string **returns** the entered number as a **string**.

Hints.

- You need to remove all the spaces from the data entered, not just the leading and trailing ones. You only need to use one string method for this.
- The solution involves using a loop. See the loops presentations for examples that can be used as the basis of a solution for this.
- Ignoring blanks lines and docstring, the solution is about 9 lines of code.
- Not all the test data used is shown.

Question 5

Create a function called **add_edit_contact**

add_edit_contact has one parameter

- contacts

contacts is a dictionary of contacts.

add_edit_contact must obtain the name of a contact by **calling** the **get_name** function.

If the entered contact name returned by get_name is not in the contacts dictionary then add_edit_contact must obtain the phone number of the contact by **calling** the **get_number_as_string** function and **passing** it the 'Contact number: ' message as the prompt. The minimum number of digits for the number (required by get_number_as_string) is 3. The add_edit_contact function then adds the contact and their phone number that was returned by get_number_as_string to the contacts dictionary and informs the user that it has been added. The phone number in the dictionary is a string.

If the entered contact returned by get_name is in the contacts dictionary then add_edit_contact informs the user that the contact is present and asks if the user would like to change the number. The add_edit_contact function **calls** the **get_option** function **passing** it 'Would you like to change the phone number? y/n: ' as the prompt to be displayed. If the user enters 'y' (case insensitive) then add_edit_contact asks the user to enter the new number for the contact by **calling** the **get_number_as_string**function and **passing** it the "New number: " message as the prompt. If the new number is different from the existing number, the add_edit_contact function then changes the phone number for the contact and informs the user that it has been changed. If the new number is the same as the existing number of the contact, add_edit_contact informs the user that the number is the same and so it won't be changed. If the user did not enter 'y' (case insensitive) then add_edit_contact informs them that the contact has not been updated.

add_edit_contact returns boolean True if the contact and or their number was added or changed, otherwise it returns False.

Hints.

- The solution is about 21/22 lines of code.
- The add_edit_contact function does not call the input function itself, that's the job of the other functions.
- The message 'Digits only please.' is generated by the get_number_as_string function.
- Don't forget to check the return value.
- Not all of the test data used is shown.

Only post the add_edit_contact function below. Do **not** include the get_name or get_number_as_string or get_option functions.

Question 6

Create a function called **remove_contact**

remove_contact has one parameter

- contacts

contacts is a dictionary of contacts.

If there are no contacts in the contact dictionary, remove_contact informs the user that the contacts dictionary is empty.

If there is a contact in the contacts dictionary, remove_contact asks the user to enter the name of a contact by **calling** the **get_name** function passing it the prompt 'Enter the contact to be removed (or press Enter to cancel): '.

If the user just presses the Enter key when asked for the contact that is to be removed, inform the user that the request is being cancelled.

If the entered contact is in the contacts dictionary then remove_contact removes the user from the contacts dictionary and informs the user that the contact has been removed.

If the entered contact (something was entered as the contact name other than the Enter key) is not in the contacts dictionary then remove_contact informs the user that the entered contact is not present.

remove_contact returns boolean True if the contact was removed, otherwise it returns False.

Hints.

- Ignoring the docstring and any comments, the solution is about 13 lines of code.
- The remove_contact function does not call the input function itself, that's the job of the get_name function.
- Not all the test data used is shown.

Only post the remove_contact function below. Do not include the get_name function.

Question 7

Create a function called **show_contact**

show_contact has one parameter

- contacts

contacts is a dictionary of contacts.

If the contacts dictionary is not empty, show_contact asks the user to enter the name of a contact by **calling** the **get_name**function.

If the entered contact returned by get_name is in the contacts dictionary then show_contact displays the phone number of the contact.

If the entered contact returned by get_name is not in the contacts dictionary then show_contact informs the user that the contact is unknown.

If the contacts dictionary is empty, show_contact informs the user that the contacts are empty.

show_contact returns boolean True if the contact was present, otherwise it returns False.

Hints.

- Ignoring the docstring and any comments, the solution is about 12 lines of code.
- The show_contact function does not call the input function itself, that's the job of the get_name function.
- Not all the test data used is shown.

Only post the show_contact function below. Do not include the get_name function.

Question 8

Create a function called **display_contacts**

display_contacts has one parameter

- contacts

contacts is a dictionary of contacts.

If the contacts dictionary is not empty then display_contacts displays the contacts and associated phone numbers as per the examples below.

If the contacts dictionary is empty then display_contacts informs the user that there are no contacts to be displayed.

list_contacts returns boolean True if the contacts were displayed, otherwise it returns False.

Hints.

- The solution is similar to a question in the lists lab. Use the code from that function as the basis (but remembering that question was referencing a list) for this one.
- Ignoring the docstring and comments, the solution is about 8 lines of code.
- Not all the test data used is shown.

Question 9

Create a function called **empty_contacts**

empty_contacts has one parameter

- contacts

contacts is a dictionary of contacts.

If the contacts dictionary is not empty and the user confirms they want to empty the contacts dictionary (by entering 'y' in either case) then empty_contacts removes all the contacts from the dictionary and **returns True**.

If the contacts dictionary is not empty and the user does not confirm they want to empty the contacts dictionary then empty_contacts informs the user accordingly and **returns False**.

If the contacts dictionary is empty then empty_contacts informs the user and **returns False**.

The empty_contacts function **must call** or invoke the **get_option** function to ascertain if the user wants to empty the contacts.

Hints.

- The solution is similar to the question raised in the lists lab. Use the code from that function as the basis for this one.
- The empty_contacts function does not call the input function itself, that's the job of the get_option function.
- Ignoring the docstring and comments, the solution is about 11 lines of code.
- Not all the test data used is shown.

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