In [13]:
```python
# Question number 1
# Write and test a function that takes a string as a parameter and returns a sorted list
# of all the unique letters used in the string. So, if the string is cheese, the list
# returned should be ['c', 'e', 'h', 's'].

# Function to get the string from the user
def get_string():
    # Prompt the user to input a string
    string= input("Enter a string of choice: ")
    # Return the entered string
    return string

# Function to process the string and return sorted unique letters
def for_sorted_string():
    # Get the string using the get_string function
    string= get_string()

    # Convert the string to a set to remove duplicates, then sort the characters
    new_string= sorted(set(string))

    # Print the sorted list of unique letters
    print(f"sorted list = {new_string}")

# Call the for_sorted_string function to execute the process
for_sorted_string()
```

```
sorted list = ['a', 'e', 'l', 'm', 'o', 'r', 't', 'w']
```

In [5]:
```python
# Question number:2
# Write and test three functions that each take two words (strings) as parameters and
# return sorted lists (as defined above) representing respectively:
# Letters that appear in at least one of the two words.
# Letters that appear in both words.
# Letters that appear in either word, but not in both.


# Function that returns letters appearing in at least one of the two words
def at_least_one (first_word, second_word):
    # Combine the letters from both words using the union (|) operator, and sort the result
    least_one = sorted(set(first_word)|set(second_word))
    # Return the sorted list of letters
    return least_one

# Function that returns letters appearing in both words
def appear_in_both (first_word, second_word):
     # Find the common letters in both words using the intersection (&) operator, and sort the result
    in_both = sorted(set(first_word)&set(second_word))
    # Return the sorted list of common letters
    return in_both

# Function that returns letters that appear in either word, but not in both
def appear_in_either_word (first_word, second_word):
     # Find the symmetric difference (^) between the sets of letters, and sort the result
     in_either_word= sorted(set(first_word)^set(second_word))
    # Return the sorted list of letters that are unique to each word
     return in_either_word

# Get input from the user for the two words
first_word= input("Enter your first word: ")
second_word= input("Enter your second word: ")

# Call each function and display the results
print(f"Letters in at least one of the two words: {at_least_one(first_word, second_word)}")
print(f"Letters in both words: {appear_in_both(first_word, second_word)}")
print(f"Letters in either word but not both: {appear_in_either_word(first_word, second_word)}")
```

```
Letters in at least one of the two words: ['a', 'b', 'd', 'e', 'g', 'i', 'l', 'm', 'n', 'o', 'r', 't', 'u', 'w']
Letters in both words: ['e', 'o', 'r', 't']
Letters in either word but not both: ['a', 'b', 'd', 'g', 'i', 'l', 'm', 'n', 'u', 'w']
```

In [4]:
```python
# Function to manage the list of countries and their capital cities
def for_countries_and_capitals():
    # Dictionary to store countries and their corresponding capitals
    countries_capitals = {}

    while True:
        # Ask the user to enter the name of a country
        name_country = input("Enter the name of the country: ")

        # Check if the input is empty, and if so, terminate the program
```

```
        if name_country.lower() == "":
            print("Thank You.")
            break

        # Standardize the country name by capitalizing it (e.g., "wales" becomes "Wales")
        name_country = name_country.capitalize()

        # Check if the country is already in the dictionary
        if name_country in countries_capitals:
            # Display the capital if the country is found in the dictionary
            print(f"The capital of {name_country} is {countries_capitals[name_country]}.")

        else:
            # If the country is not found, prompt the user to enter the capital
            capital = input("Please enter the name of the capital: ")
            # Add the country and capital to the dictionary
            countries_capitals[name_country] = capital
            print(f"Thank you! I've added {name_country} with its capital {capital}.")

# Main program execution
if __name__ == "__main__":
    # Call the function to manage countries and capitals
    for_countries_and_capitals()
```

```
Thank you! I've added Nepal with its capital kathmandu .
Thank you! I've added Bhutan  with its capital thimpu.
Thank you! I've added Bangladesh with its capital dhaka.
The capital of Bangladesh is dhaka.
Thank You.
```

In [4]:

```
# Question number:4
# One approach to analysing some encrypted data where a substitution is suspected
# is frequency analysis. A count of the different symbols in the message can be used
# to identify the language used, and sometimes some of the letters. In English, the
# most common letter is "e", and so the symbol representing "e" should appear most
# in the encrypted text.
# Write a program that processes a string representing a message and reports the six
# most common letters, along with the number of times they appear. Case should
# not matter, so "E" and "e" are considered the same.
# Hint: There are many ways to do this. It is obviously a dictionary, but we will want
# zero counts, so some initialisation is needed. Also, sorting dictionaries is tricky, so
# best to ignore that initially, and then check the usual resources for the runes.

from collections import Counter

def most_common_letters(message):
    # Convert the message to lowercase
    message = message.lower()
    # Count the frequency of alphabetic characters
    letter_counts = Counter(filter(str.isalpha, message))
    # Get the six most common letters
    most_common = letter_counts.most_common(6)
    return most_common

# Get user input
message = input("Enter a message to see common letters within the message: ")

# Call the function and store the result
top_six = most_common_letters(message)

# Print the six most common letters
print("The six most common letters are:")
for letter, count in top_six:
    print(f"{letter}: {count}")
```

```
The six most common letters are:
c: 3
e: 3
r: 3
o: 2
m: 2
p: 2
```

In [ ]: