Problem 1: Palindrome Checker

Scenario: You are building a text editor that highlights palindromes in the text. A palindrome is a word that reads the same backward as forward.

Description: Write a script that checks whether a given string `s` is a palindrome or not. Ignore spaces, punctuation, and case differences.

Sample Input:
```
s = "A man, a plan, a canal, Panama"
```

Sample Output:
```
True
```

Sample Input:
```
s = "Hello, World!"
```

Sample Output:
```
False
```

Problem 2: Fibonacci Sequence

Scenario: You are developing a mathematical tool that helps students visualize the Fibonacci sequence.

Description: Write a script that generates the first `n` numbers in the Fibonacci sequence.

Sample Input:
```
n = 5
```

Sample Output:
```
[0, 1, 1, 2, 3]
```

Sample Input:
```
n = 10
```

Sample Output:

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Problem 3: Anagram Checker

Scenario: You are creating a word game where players need to form anagrams of a given word to score points.

Description: Write a script that checks if two strings `s1` and `s2` are anagrams. An anagram is a word formed by rearranging the letters of another word, using all the original letters exactly once.

Sample Input:
```
s1 = "listen"
s2 = "silent"
```

Sample Output:
```
True
```

Sample Input:
```
s1 = "hello"
s2 = "world"
```

Sample Output:
```
False
```

Problem 4: Prime Number Checker

Scenario: You are writing a utility for a number theory class that determines if a given number is prime.

Description: Write a script that checks if a given number `n` is a prime number. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Sample Input:
```
n = 7
```

Sample Output:
```
True
```

Sample Input:
```
n = 10
```

Sample Output:

```
False
```

Problem 5: Sum of Digits

Scenario: You are developing a feature for a financial application that calculates the digital root of a number, which is the sum of its digits.

Description: Write a script that calculates the sum of the digits of a given number `n`.

Sample Input:
```
n = 1234
```

Sample Output:
```
10
```

Sample Input:
```
n = 5678
```

Sample Output:
```
26
```

Problem 6: Reverse a List

Scenario: You are building a utility function for a data processing tool that needs to reverse the order of elements in a list.

Description: Write a script that reverses a given list `lst`.

Sample Input:
```
lst = [1, 2, 3, 4, 5]
```

Sample Output:
```
[5, 4, 3, 2, 1]
```

Sample Input:
```
lst = ['a', 'b', 'c']
```

Sample Output

```
['c', 'b', 'a']
```

Problem 7: Character Frequency

Scenario: You are creating a text analysis tool that provides statistics on character frequency in a given text.

Description: Write a script that counts the frequency of each character in a given string `s`.

Sample Input:

```
s = "abracadabra"
```

Sample Output:

```
{'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1}
```

Sample Input:
```
s = "hello"
```

Sample Output:
```
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

Problem 8: Merge Two Sorted Lists

Scenario: You are working on a sorting algorithm that needs to merge two pre-sorted lists into a single sorted list.

Description: Write a script that merges two sorted lists `lst1` and `lst2` into one sorted list.

Sample Input:
```
lst1 = [1, 3, 5]
lst2 = [2, 4, 6]
```

Sample Output:
```
[1, 2, 3, 4, 5, 6]
```

Sample Input:
```
lst1 = [1, 2, 3]
lst2 = [4, 5, 6]
```

Sample Output:

```
[1, 2, 3, 4, 5, 6]
```

Problem 9: Find Common Elements

Scenario: You are building a feature for a social media application that finds common friends between two users.

Description: Write a script that finds the common elements between two lists `lst1` and `lst2`.

Sample Input:
```
lst1 = [1, 2, 3, 4]
lst2 = [3, 4, 5, 6]
```

Sample Output:
```
[3, 4]
```

Sample Input:
```
lst1 = ['a', 'b', 'c']
lst2 = ['b', 'c', 'd']
```

Sample Output:
```
['b', 'c']
```

Problem 10: Remove Duplicates from List

Scenario: You are writing a data cleanup script that needs to remove duplicate entries from a list of items.

Description: Write a script that removes duplicates from a given list `lst`.

Sample Input:

```
lst = [1, 2, 2, 3, 4, 4, 5]
```

Sample Output:

```
[1, 2, 3, 4, 5]
```

Sample Input:

```
lst = ['a', 'b', 'a', 'c', 'd', 'd']
```

Sample Output:

```
['a', 'b', 'c', 'd']
```