Sure, here are the solutions to the problems:

1. Reverse a string without using any built-in string reversal functions:

```python

def reverse\_string(input\_str):

reversed\_str = ''

for char in input\_str:

reversed\_str = char + reversed\_str

return reversed\_str

```

2. Check if a given string is a palindrome:

```python

def is\_palindrome(input\_str):

return input\_str == input\_str[::-1]

```

3. Find the largest element in a given list:

```python

def find\_largest\_element(input\_list):

if not input\_list:

return None

max\_element = input\_list[0]

for num in input\_list:

if num > max\_element:

max\_element = num

return max\_element

```

4. Count the occurrence of each element in a list:

```python

def count\_occurrences(input\_list):

occurrences = {}

for element in input\_list:

occurrences[element] = occurrences.get(element, 0) + 1

return occurrences

```

5. Find the second largest number in a list:

```python

def find\_second\_largest(input\_list):

if len(input\_list) < 2:

return None

largest = second\_largest = float('-inf')

for num in input\_list:

if num > largest:

second\_largest = largest

largest = num

elif num > second\_largest and num != largest:

second\_largest = num

return second\_largest

```

6. Remove duplicate elements from a list:

```python

def remove\_duplicates(input\_list):

return list(dict.fromkeys(input\_list))

```

7. Calculate the factorial of a given number:

```python

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

```

8. Check if a given number is prime:

```python

def is\_prime(n):

if n < 2:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

```

9. Sort a list of integers in ascending order:

```python

def sort\_list(input\_list):

return sorted(input\_list)

```

10. Find the sum of all numbers in a list:

```python

def sum\_list(input\_list):

return sum(input\_list)

```

11. Find the common elements between two lists:

```python

def find\_common\_elements(list1, list2):

return list(set(list1) & set(list2))

```

12. Check if a given string is an anagram of another string:

```python

def is\_anagram(str1, str2):

return sorted(str1) == sorted(str2)

```

13. Generate all permutations of a given string:

```python

from itertools import permutations

def generate\_permutations(input\_str):

return [''.join(p) for p in permutations(input\_str)]

```

14. Calculate the Fibonacci sequence up to a given number of terms:

```python

def fibonacci(n):

fib\_sequence = [0, 1]

while len(fib\_sequence) < n:

fib\_sequence.append(fib\_sequence[-1] + fib\_sequence[-2])

return fib\_sequence

```

15. Find the median of a list of numbers:

```python

def find\_median(input\_list):

sorted\_list = sorted(input\_list)

n = len(sorted\_list)

if n % 2 == 1:

return sorted\_list[n // 2]

else:

return (sorted\_list[n // 2 - 1] + sorted\_list[n // 2]) / 2

```

16. Check if a given list is sorted in non-decreasing order:

```python

def is\_sorted(input\_list):

return all(input\_list[i] <= input\_list[i+1] for i in range(len(input\_list)-1))

```

17. Find the intersection of two lists:

```python

def find\_intersection(list1, list2):

return list(set(list1) & set(list2))

```

18. Find the maximum subarray sum in a given list:

```python

def max\_subarray\_sum(input\_list):

max\_sum = current\_sum = input\_list[0]

for num in input\_list[1:]:

current\_sum = max(num, current\_sum + num)

max\_sum = max(max\_sum, current\_sum)

return max\_sum

```

19. Remove all vowels from a given string:

```python

def remove\_vowels(input\_str):

vowels = "aeiouAEIOU"

return ''.join(char for char in input\_str if char not in vowels)

```

20. Reverse the order of words in a given sentence:

```python

def reverse\_words(sentence):

words = sentence.split()

reversed\_sentence = ' '.join(reversed(words))

return reversed\_sentence

```

21. Check if two strings are anagrams of each other:

```python

def are\_anagrams(str1, str2):

return sorted(str1) == sorted(str2)

```

22. Find the first non-repeating character in a string:

```python

def first\_non\_repeating\_character(input\_str):

char\_count = {}

for char in input\_str:

char\_count[char] = char\_count.get(char, 0) + 1

for char in input\_str:

if char\_count[char] == 1:

return char

return None

```

23. Find the prime factors of a given number:

```python

def prime\_factors(n):

factors = []

divisor = 2

while n > 1:

if n % divisor == 0:

factors.append(divisor)

n //= divisor

else:

divisor += 1

return factors

```

24. Check if a given number is a power of two:

```python

def is\_power\_of\_two(n):

return n > 0 and (n & (n - 1) == 0)

```

25. Merge two sorted lists into a single sorted list:

```python

def merge\_sorted\_lists(list1, list2):

merged\_list = []

i = j = 0

while i < len(list1) and j < len(list2):

if list1[i] < list2[j]:

merged\_list.append(list1[i])

i += 1

else:

merged\_list.append(list2[j])

j += 1

merged\_list.extend(list1[i:])

merged\_list.extend(list2[j:])

return merged\_list

```

26. Find the mode of a list of numbers:

```python

from collections import Counter

def find\_mode(input\_list):

count\_dict = Counter(input\_list)

max\_count = max(count\_dict.values())

return [num for num, count in count\_dict.items() if count == max\_count]

```

27. Find the greatest common divisor (GCD) of two numbers:

```python

def gcd(a, b):

while b:

a, b = b, a % b

return a

```

28. Calculate the square root of a given number:

```python

def square\_root(n):

return n \*\* 0.5

```

29. Check if a given string is a valid palindrome

ignoring non-alphanumeric characters:

```python

import re

def is\_valid\_palindrome(input\_str):

clean\_str = re.sub(r'[^a-zA-Z0-9]', '', input\_str)

return clean\_str.lower() == clean\_str[::-1].lower()

```

30. Find the minimum element in a rotated sorted list:

```python

def find\_minimum\_rotated\_sorted(input\_list):

left, right = 0, len(input\_list) - 1

while left < right:

mid = (left + right) // 2

if input\_list[mid] > input\_list[right]:

left = mid + 1

else:

right = mid

return input\_list[left]

```

31. Find the sum of all even numbers in a list:

```python

def sum\_even\_numbers(input\_list):

return sum(num for num in input\_list if num % 2 == 0)

```

32. Calculate the power of a number using recursion:

```python

def power(base, exponent):

if exponent == 0:

return 1

elif exponent > 0:

return base \* power(base, exponent - 1)

else:

return 1 / (base \* power(base, -exponent - 1))

```

33. Remove duplicates from a list while preserving the order:

```python

def remove\_duplicates\_preserve\_order(input\_list):

unique\_list = []

for item in input\_list:

if item not in unique\_list:

unique\_list.append(item)

return unique\_list

```

34. Find the longest common prefix among a list of strings:

```python

def longest\_common\_prefix(str\_list):

if not str\_list:

return ''

prefix = str\_list[0]

for string in str\_list[1:]:

i = 0

while i < len(prefix) and i < len(string) and prefix[i] == string[i]:

i += 1

prefix = prefix[:i]

return prefix

```

35. Check if a given number is a perfect square:

```python

def is\_perfect\_square(n):

root = int(n \*\* 0.5)

return root \* root == n

```

36. Calculate the product of all elements in a list:

```python

def product\_list(input\_list):

result = 1

for num in input\_list:

result \*= num

return result

```

37. Reverse the order of words in a sentence while preserving the word order:

```python

def reverse\_words\_preserve\_order(sentence):

words = sentence.split()

reversed\_sentence = ' '.join(reversed(words))

return reversed\_sentence

```

38. Find the missing number in a given list of consecutive numbers:

```python

def find\_missing\_number(input\_list):

n = len(input\_list) + 1

expected\_sum = n \* (n + 1) // 2

actual\_sum = sum(input\_list)

return expected\_sum - actual\_sum

```

39. Find the sum of digits of a given number:

```python

def sum\_of\_digits(n):

return sum(int(digit) for digit in str(abs(n)))

```

40. Check if a given string is a valid palindrome considering case sensitivity:

```python

def is\_valid\_palindrome\_case\_sensitive(input\_str):

return input\_str == input\_str[::-1]

```

41. Find the smallest missing positive integer in a list:

```python

def smallest\_missing\_positive(input\_list):

n = len(input\_list)

for i in range(n):

while 1 <= input\_list[i] <= n and input\_list[input\_list[i] - 1] != input\_list[i]:

input\_list[input\_list[i] - 1], input\_list[i] = input\_list[i], input\_list[input\_list[i] - 1]

for i in range(n):

if input\_list[i] != i + 1:

return i + 1

return n + 1

```

42. Find the longest palindrome substring in a given string:

```python

def longest\_palindrome\_substring(input\_str):

longest = ""

for i in range(len(input\_str)):

for j in range(i, len(input\_str)):

substring = input\_str[i:j+1]

if substring == substring[::-1] and len(substring) > len(longest):

longest = substring

return longest

```

43. Find the number of occurrences of a given element in a list:

```python

def count\_occurrences\_element(input\_list, target):

return input\_list.count(target)

```

44. Check if a given number is a perfect number:

```python

def is\_perfect\_number(n):

divisors = [i for i in range(1, n) if n % i == 0]

return sum(divisors) == n

```

45. Remove all duplicates from a string:

```python

def remove\_duplicates\_string(input\_str):

return ''.join(dict.fromkeys(input\_str))

```

46. Find the first missing positive integer in a list:

```python

def first\_missing\_positive(input\_list):

n = len(input\_list)

for i in range(n):

while 1 <= input\_list[i] <= n and input\_list[input\_list[i] - 1] != input\_list[i]:

input\_list[input\_list[i] - 1], input\_list[i] = input\_list[i], input\_list[input\_list[i] - 1]

for i in range(n):

if input\_list[i] != i + 1:

return i + 1

return n + 1

```

Note: The solutions provided above do not include any test cases. It is essential to test the functions with various input scenarios to ensure their correctness.