

1.Reverse number

```
def reverse_number(num, rev=0):  
    if num == 0:  
        return rev  
    else:  
        return reverse_number(num // 10, rev * 10 + num % 10)  
num = 12345  
print(reverse_number(num))
```

2.Perfect number

```
def is_perfect(n):  
    sum_divisors = 0  
    for i in range(1, n // 2 + 1):  
        if n % i == 0:  
            sum_divisors += i  
    return sum_divisors == n  
n = 28  
print(is_perfect(n))
```

3.Demonstrate Usage of Big-O Notation

```
def constant_time(n):  
    return n + 1  
def linear_time(arr):  
    total = 0  
    for num in arr:  
        total += num  
    return total  
  
def quadratic_time(arr):  
    for i in range(len(arr)):  
        for j in range(len(arr)):  
            print(i, j)  
print(constant_time(5))  
print(linear_time([1, 2, 3, 4, 5]))  
quadratic_time([1, 2, 3])
```

4.Mathematical Analysis of Non-Recursive and Recursive Algorithms

```
def linear_search(arr, target):
```

```

for i in range(len(arr)):
    if arr[i] == target:
        return i
return -1
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(linear_search([1, 2, 3, 4, 5], 4))
print(factorial(5))

```

5.Solving Recurrence relations

```

def master_theorem(n):
    if n == 1:
        return 1
    return 2 * master_theorem(n // 2) + n
print(master_theorem(8))

```

```

def substitution_method(n):
    if n == 1:
        return 1
    return substitution_method(n - 1) + 1
print(substitution_method(5))

```

```

def iteration_method(n):
    if n == 1:
        return 1
    return 2 * iteration_method(n // 2) + n

```

6.Intersection unique

```

def intersection_unique(nums1, nums2):
    return list(set(nums1) & set(nums2))
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersection_unique(nums1, nums2))

```

```
from collections import Counter
```

7.intersection multiset

```
def intersection_multiset(nums1, nums2):  
  
    counts1 = Counter(nums1)  
  
    counts2 = Counter(nums2)  
  
    intersection = []  
  
    for num in counts1:  
  
        if num in counts2:  
  
            intersection.extend([num] * min(counts1[num], counts2[num]))  
  
    return intersection  
  
nums1 = [1, 2, 2, 1]  
  
nums2 = [2, 2]  
  
print(intersection_multiset(nums1, nums2))
```

8.merge sort

```
def merge_sort(arr):  
  
    if len(arr) > 1:  
  
        mid = len(arr) // 2  
  
        L = arr[:mid]  
  
        R = arr[mid:]  
  
        merge_sort(L)  
  
        merge_sort(R)  
  
        i = j = k = 0  
  
        while i < len(L) and j < len(R):  
  
            if L[i] < R[j]:  
  
                arr[k] = L[i]  
  
                i += 1  
  
            else:  
  
                arr[k] = R[j]  
  
                j += 1  
  
            k += 1  
  
        while i < len(L):  
  
            arr[k] = L[i]  
  
            i += 1  
  
            k += 1  
  
        while j < len(R):  
  
            arr[k] = R[j]
```

```

        j += 1

        k += 1

    return arr

nums = [5, 2, 3, 1]
print(merge_sort(nums))

```

9.sort half_odd and half_even

```

def sort_half_odd_half_even(nums):

    odd_index = 0

    even_index = 1

    n = len(nums)

    while odd_index < n and even_index < n:

        if nums[odd_index] % 2 == 0:

            while even_index < n and nums[even_index] % 2 == 0:

                even_index += 2

            if even_index < n:

                nums[odd_index], nums[even_index] = nums[even_index], nums[odd_index]

            odd_index += 2

    return nums

nums = [4, 1, 2, 3, 6, 7, 8, 5]

print(sort_half_odd_half_even(nums))

```

10.sorted array

```

def sort_array_by_parity(nums):

    odd_index = 1

    even_index = 0

    n = len(nums)

    while odd_index < n and even_index < n:

        if nums[even_index] % 2 == 0:

            even_index += 2

        elif nums[odd_index] % 2 == 1:

            odd_index += 2

        else:

            nums[even_index], nums[odd_index] = nums[odd_index], nums[even_index]

            even_index += 2

            odd_index += 2

    return nums

nums = [4, 1, 2, 3, 6, 7, 8, 5]

print(sort_array_by_parity(nums))

```

