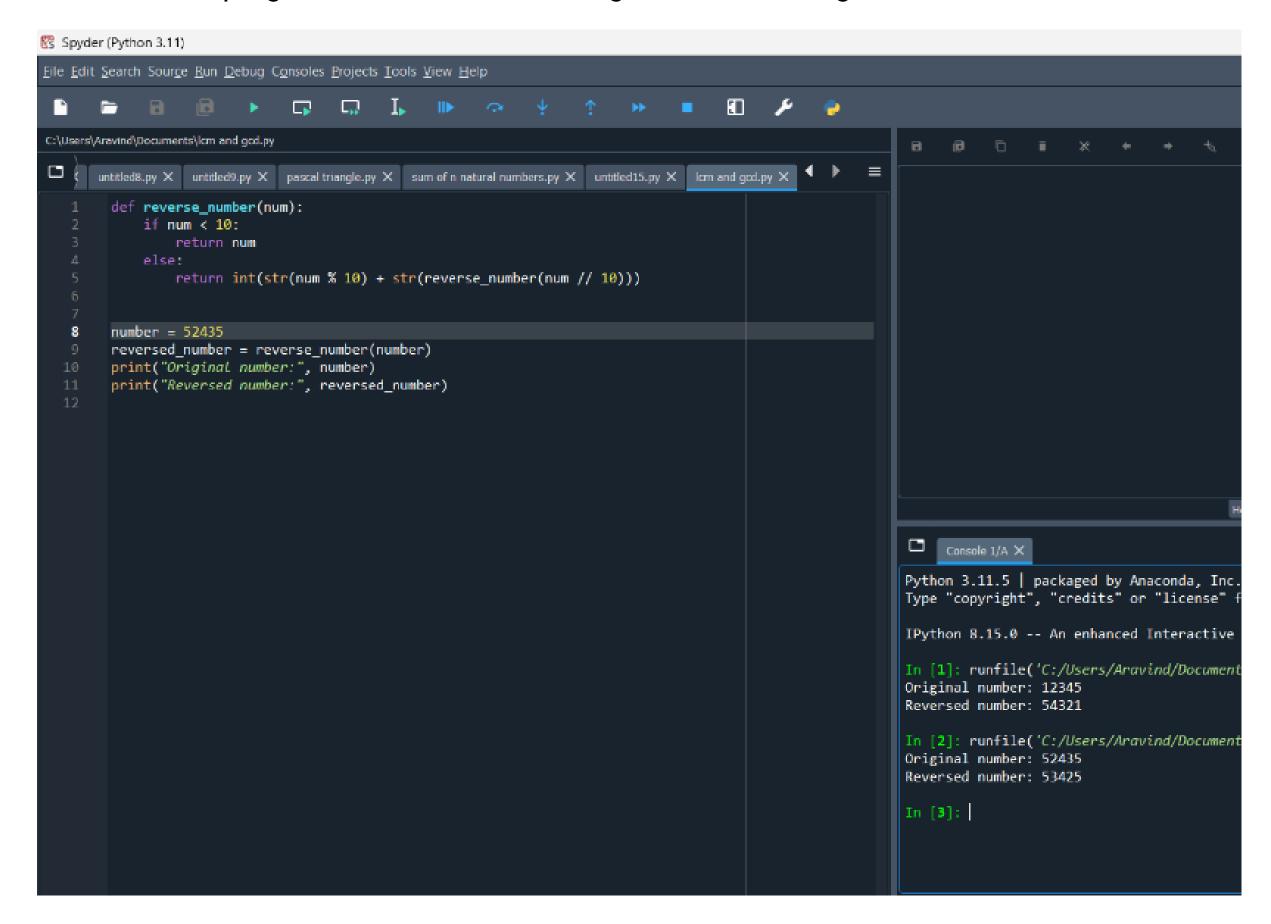
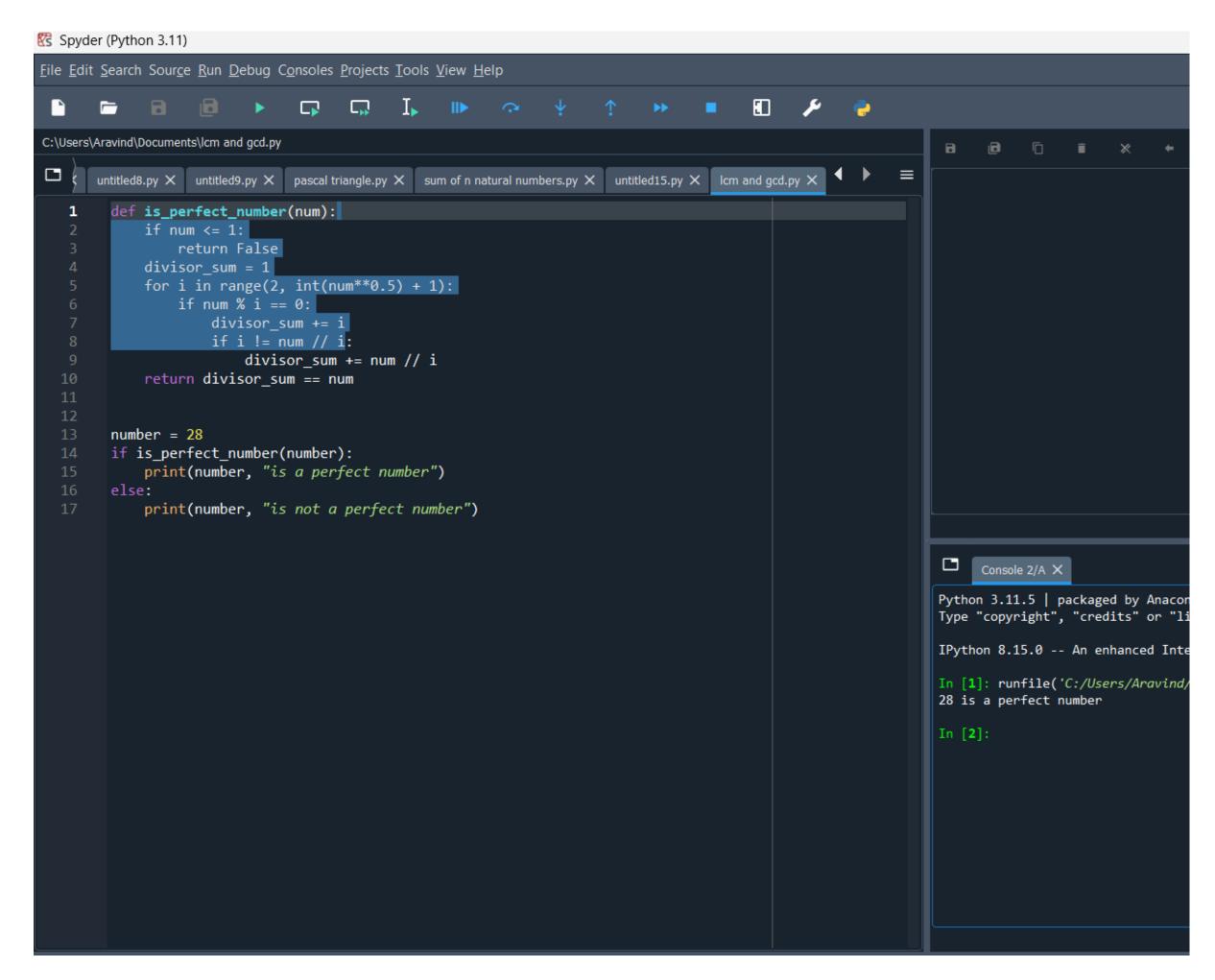
1. Write a program to find the reverse of a given number using recursive.



2. Write a program to find the perfect number



3. Write C program that demonstrates the usage of these notations by analyzing the time complexity of some example algorithms.

```
'pythnon.py - C;/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py (3.12.1)*
File Edit Format Run Options Window Help
def constant algo(items):
                                                                           IDLE Shell 3.12.1
                                                                                                                                                           result = items[0] * items[0]
    print(result)
                                                                           Eile Edit Shell Debug Options Window Help
                                                                              = RESTART: C:/Users/Aravind/AppData/Local/Programs/Fython/Python312/pythnon.py
def linear algo(items):
                                                                              Sorted array: [7, 56, 9, 4, 7, 2]
    for item in items:
        print(item)
                                                                              - RESTART: C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py -
def quadratic algo(items):
    for item in items:
        for i in range(len(items)):
            print(item, items[i])
def logarithmic_algo(n):
   while n > 1:
        print(n)
        n = n // 2
def exponential_algo(n):
   if n <- 1:
        return n
                                                                              2 3
2 4
2 5
3 1
        return exponential_algo(n-1) + exponential_algo(n-2)
items = [1, 2, 3, 4, 5]
constant_algo(items)
                                                                              3 3
linear_algo(items)
quadratic algo(items)
                                                                              3 5
logarithmic_algo(16)
                                                                              4 1
exponential_algo(5)
                                                                              4 2
                                                                              4.5
                                                                              5 1
                                                                              5 3
                                                                              5 4
5 5
                                                                              16
                                                                              8
                                                                              4
                                                                           >>>
                                                                                                                                                           Ln: 43 Col: 0
```

4. Write C programs that demonstrate the mathematical analysis of non-recursive and recursive algorithms

```
pythnon.py - C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py (3.12.1)
File Edit Format Run Options Window Help
def non_recursive_factorial(n):
   result = 1
                                                                              IDLE Shell 3.12.1
                                                                                                                                                             for i in range(1, n + 1):
       result *- i
                                                                              File Edit Shell Debug Options Window Help
    return result
                                                                                 Python 3.12.1 (tags/v3.12.1:2305ca5, Dec. 7 2023, 22:03:25) [MSC v.1937 64 bit ( *
                                                                                 AMD64)] on win32
def recursive_factorial(n):
                                                                                 Type "help", "copyright", "credits" or "license()" for more information.
    if n -- 0:
       return 1
                                                                                 = RESTART: C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py
                                                                                 Non-Recursive Factorial of 5: 120
        return n * recursive_factorial(n - 1)
                                                                                 Recursive Factorial of 5: 120
print(f"Non-Recursive Factorial of (n): (non_recursive_factorial(n))")
print(f"Recursive Factorial of (n): (recursive_factorial(n))")
                                                                                                                                                              Ln: 7 Col: 0
```

5. Write C programs for solving recurrence relations using the Master Theorem, Substitution Method, and Iteration Method will demonstrate how to calculate the time complexity of an example recurrence relation using the specified technique.

```
pythnon.py - C;/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py (3.12.1)
File Edit Format Run Options Window Help
def master theorem(a, b, k):
    if a > b ** k:
                                                         IDLE Shell 3.12.1
        return "O(n^log_b(a))"
    elif a == b ** k:
                                                         File Edit Shell Debug Options Window Help
       return "O(n^k * log(n))"
                                                            Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit ( *
                                                            AMD64)] on win32
        return "O(n^k)"
                                                            Type "help", "copyright", "credits" or "license()" for more information.
result = master theorem(3, 2, 1)
                                                            = RESTART: C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py
print(result)
                                                            O(n^log b(a))
                                                        >>>
                                                                                                                                          Ln: 6 Col: 0
```

6. Given two integer arrays nums 1 and nums 2, return an array of their Intersection. Each element in the result must be unique and you may return the result in any order.

```
pythnon.py - C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py (3.12.1)
Eile Edit Format Run Options Window Help
def intersection(nums1, nums2):
    set1 = set(nums1)
                                                                3.12.1 IDLE Shell 3.12.1
    set2 = set(nums2)
    return list(set1.intersection(set2))
                                                                Eile Edit Shell Debug Options Window Help
                                                                   Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (
                                                                    AMD64)] on win32
nums1 = [1, 2, 2, 1]
                                                                    Type "help", "copyright", "credits" or "license()" for more information.
nums2 = [2, 2]
print(intersection(nums1, nums2)) # Output: [2]
                                                                    = RESTART: C:/Users/Aravind/AppData/Local/Frograms/Fython/Fython312/pythnon.py
                                                                    = RESTART: C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py =
                                                                >>>
                                                                                                                                                  Ln: 8 Col: 0
```

7. Given two integer arrays nums 1 and nums 2, return an array of their intersection. Each element in the result must appear as many times as it shows in both arrays and you may return the result in any order.

```
pythnon.py - C/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py (3.12.1)
File Edit Format Bun Options Window Help
from collections import Counter
                                                                              IDLE Shell 3.12.1
det intersect(nums1, nums2):
                                                                             File Edit Shell Debug Options Window Help
    count1, count2 = Counter(nums1), Counter(nums2)
                                                                                 Python 3.12.1 (tags/v3.12.1:2305ca5, Dec. 7 2023, 22:03:25) [MSC v.1937 64 bit ( *
    intersection = []
                                                                                 AMD64)] on win32
    for num in countl.keys() & count2.keys():
                                                                                 Type "help", "copyright", "credits" or "license()" for more information.
        intersection.extend([num] * min(count1[num], count2[num]))
    return intersection
                                                                                 - RESTART: C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythmon.py
# Example Usage
nums1 = [1, 2, 2, 3, 1]
num52 = [2, 3]
print(intersect(nums1, nums2)) # Output: [2, 2]
                                                                                                                                                              Lm 6 Colt 0
```

8. Given an array of integers nums, sort the array in ascending order and return it. You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

```
🌛 pythnor.py - C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnor.py (3.12.1)
Eile Edit Format Run Options Window Help
def merge_sort(nums):
                                                                             IDLE Shell 3.12.1
                                                                                                                                                            \times
   if len(nums) <= 1:
        return nums
                                                                             File Edit Shell Debug Options Window Help
                                                                                Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (
    mid = len(nums) // 2
                                                                                AMD64)] on win32
    left = merge_sort(nums[:mid])
                                                                               Type "help", "copyright", "credits" or "license()" for more information.
    right - merge_sort(nums[mid:])
                                                                                = RESTART: C:/Users/Aravind/AppData/Local/Programs/Fython/Python312/pythnon.py
    return merge(left, right)
def merge(left, right):
                                                                            >>>
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:</pre>
            result.append(left[i])
        else:
            result.append(right[j])
            j += 1
    result.extend(left[i:])
    result.extend(right[j:])
    return result
nums = [12, 4, 7, 1, 9, 3]
sorted nums - merge sort(nums)
print(sorted_nums)
                                                                                                                                                             Ln: 7 Col: 0
```

9. Given an array of integers nums, half of the integers in nums are odd, and the other half are even.

```
bythnon.py - C/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py (3.12.1)
Eile Edit Format Bun Options Window Help
def divide array(nums):
                                                                     IDLE Shell 3.12.1
   odd nums = [num for num in nums if num % 2 != 0]
    even nums = [num for num in nums if num % 2 == 0]
                                                                        Fython 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit ( *
    return odd nums, even nums
                                                                        AMD64)] on win32
nums = [1, 2, 3, 4, 5, 6]
                                                                         Type "help", "copyright", "credits" or "license()" for more information.
odd nums, even nums = divide array(nums)
print("Odd Numbers:", odd nums)

    RESTART: C:/Users/Aravind/AppData/Local/Programs/Python/Python312/pythnon.py

print("Even Numbers:", even nums)
                                                                        Odd Numbers: [1, 3, 5]
                                                                        Even Numbers: [2, 4, 6]
                                                                                                                                                      Ln: 8 Col: 0
```

10. Sort the array so that whenever nums[i] is odd, i is odd, and whenever nums[i] is even, i is even. Return any answer array that satisfies this condition.

Ln: 6 Col: 0