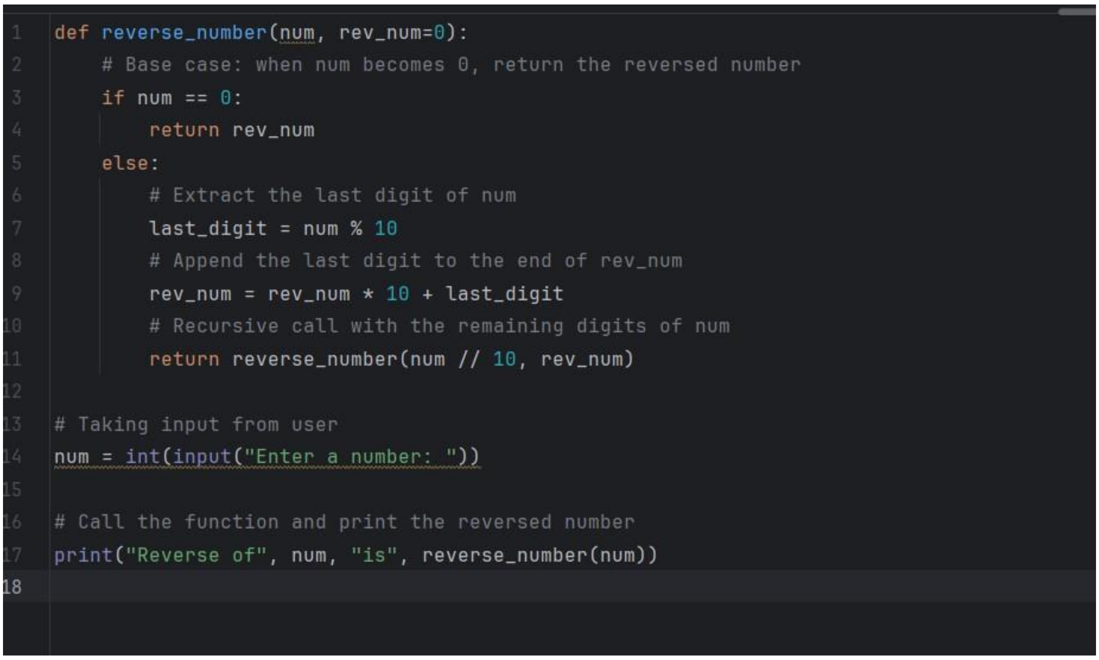
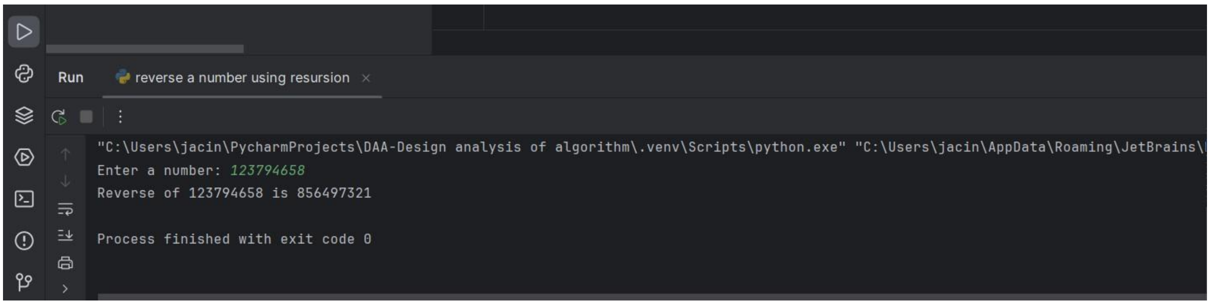
Lab Program -2

Date:05/06/24

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

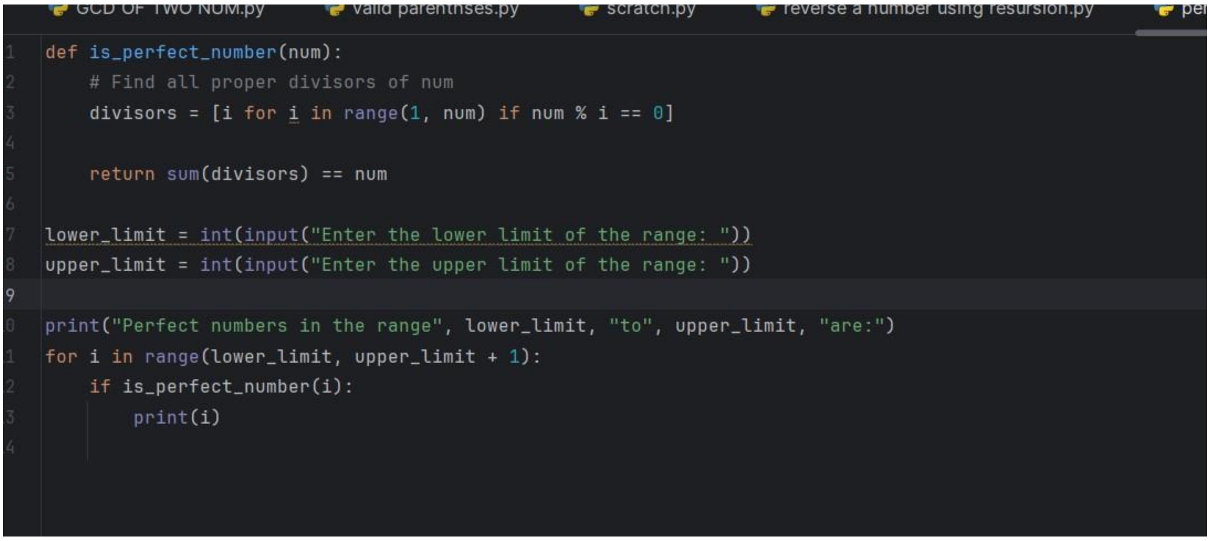


```python
def reverse_number(num, rev_num=0):
    # Base case: when num becomes 0, return the reversed number
    if num == 0:
        return rev_num
    else:
        # Extract the last digit of num
        last_digit = num % 10
        # Append the last digit to the end of rev_num
        rev_num = rev_num * 10 + last_digit
        # Recursive call with the remaining digits of num
        return reverse_number(num // 10, rev_num)

# Taking input from user
num = int(input("Enter a number: "))

# Call the function and print the reversed number
print("Reverse of", num, "is", reverse_number(num))

```

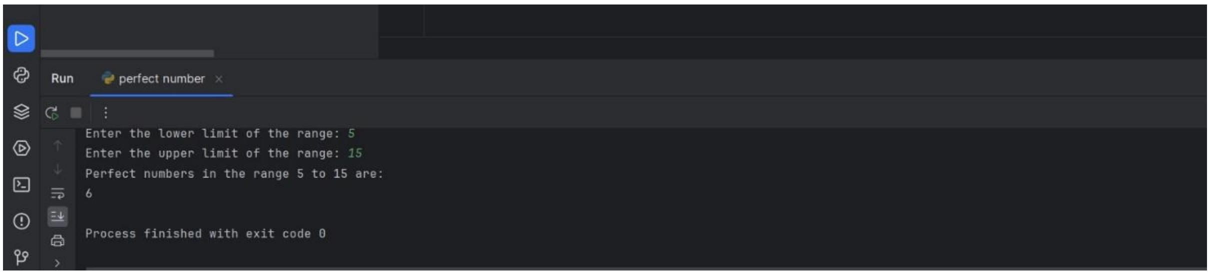


```
"C:\Users\jacin\PycharmProjects\DAA-Design analysis of algorithm\.venv\Scripts\python.exe" "C:\Users\jacin\AppData\Roaming\JetBrains\
Enter a number: 123794658
Reverse of 123794658 is 856497321

Process finished with exit code 0
```

2. Write a program to find the perfect number.



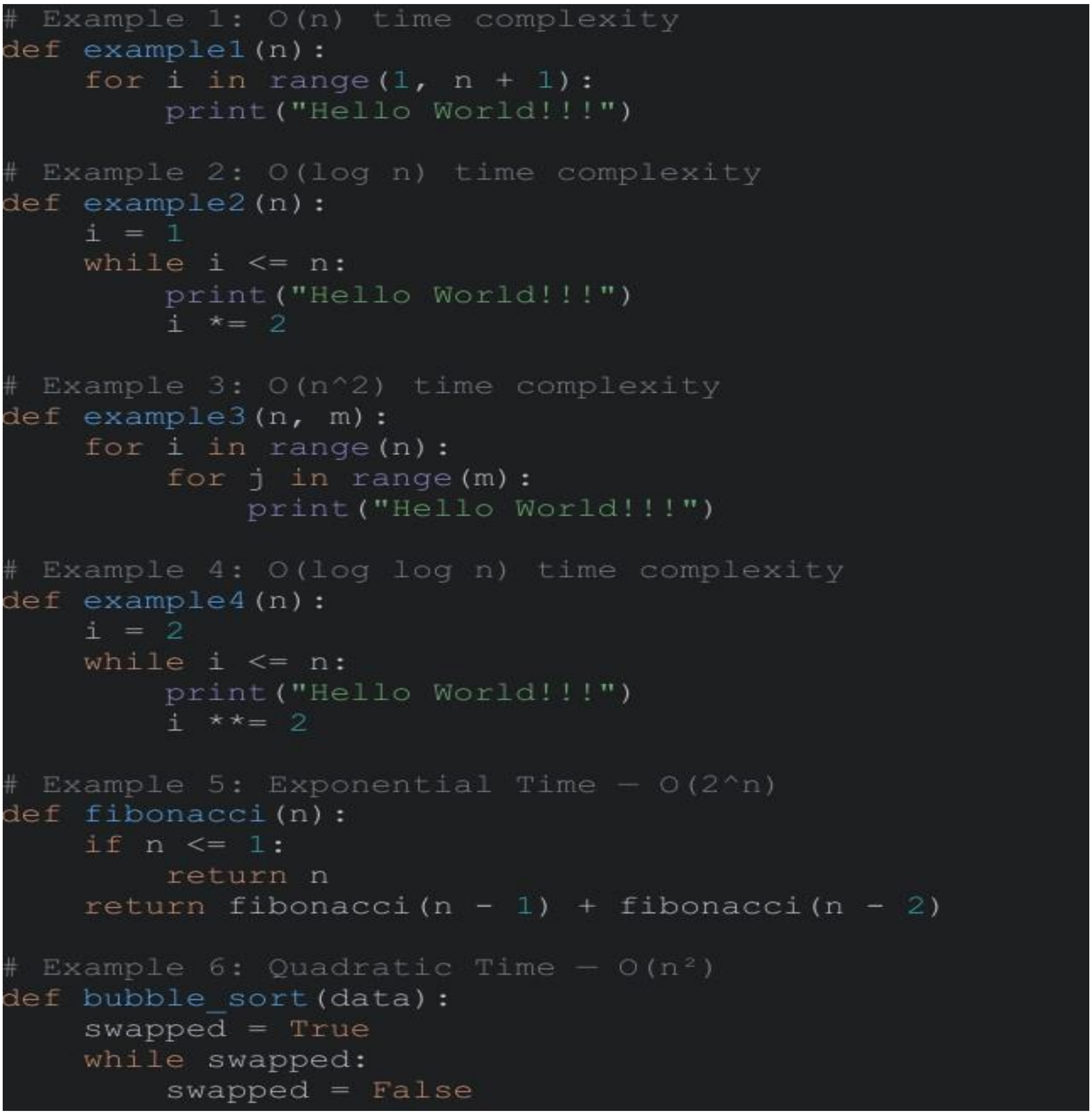
```python
def is_perfect_number(num):
    # Find all proper divisors of num
    divisors = [i for i in range(1, num) if num % i == 0]

    return sum(divisors) == num

lower_limit = int(input("Enter the lower limit of the range: "))
upper_limit = int(input("Enter the upper limit of the range: "))

print("Perfect numbers in the range", lower_limit, "to", upper_limit, "are:")
for i in range(lower_limit, upper_limit + 1):
    if is_perfect_number(i):
        print(i)
```



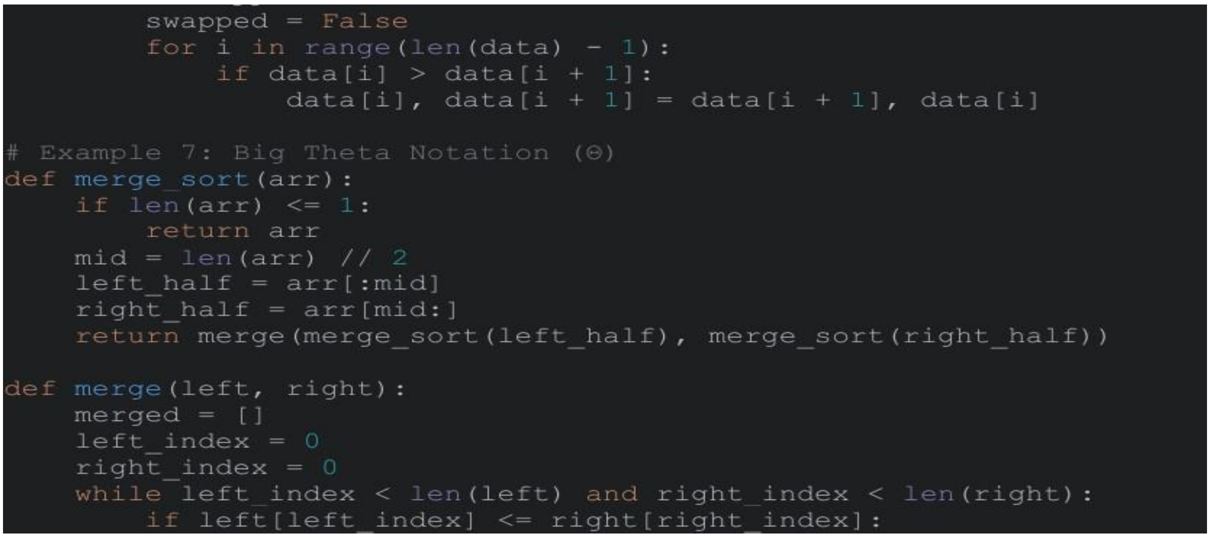
```
Enter the lower limit of the range: 5
Enter the upper limit of the range: 15
Perfect numbers in the range 5 to 15 are:
6

Process finished with exit code 0
```

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

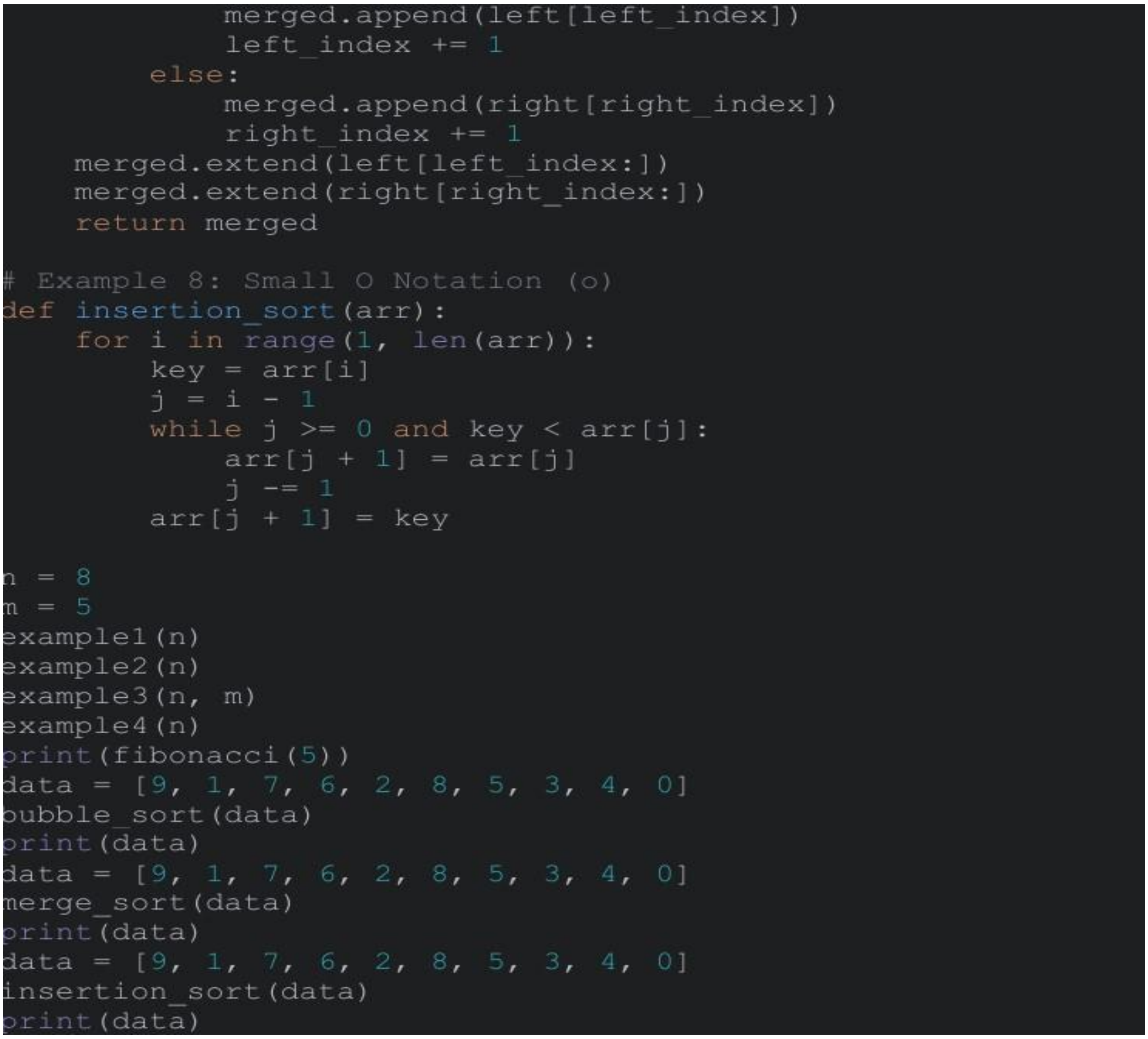
   coding



```
# Example 1: O(n) time complexity
def example1(n):
    for i in range(1, n + 1):
        print("Hello World!!!")

# Example 2: O(log n) time complexity
def example2(n):
    i = 1
    while i <= n:
        print("Hello World!!!")
        i *= 2

# Example 3: O(n^2) time complexity
def example3(n, m):
    for i in range(n):
        for j in range(m):
            print("Hello World!!!")

# Example 4: O(log log n) time complexity
def example4(n):
    i = 2
    while i <= n:
        print("Hello World!!!")
        i **= 2

# Example 5: Exponential Time — O(2^n)
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

# Example 6: Quadratic Time — O(n²)
def bubble_sort(data):
    swapped = True
    while swapped:
        swapped = False
```

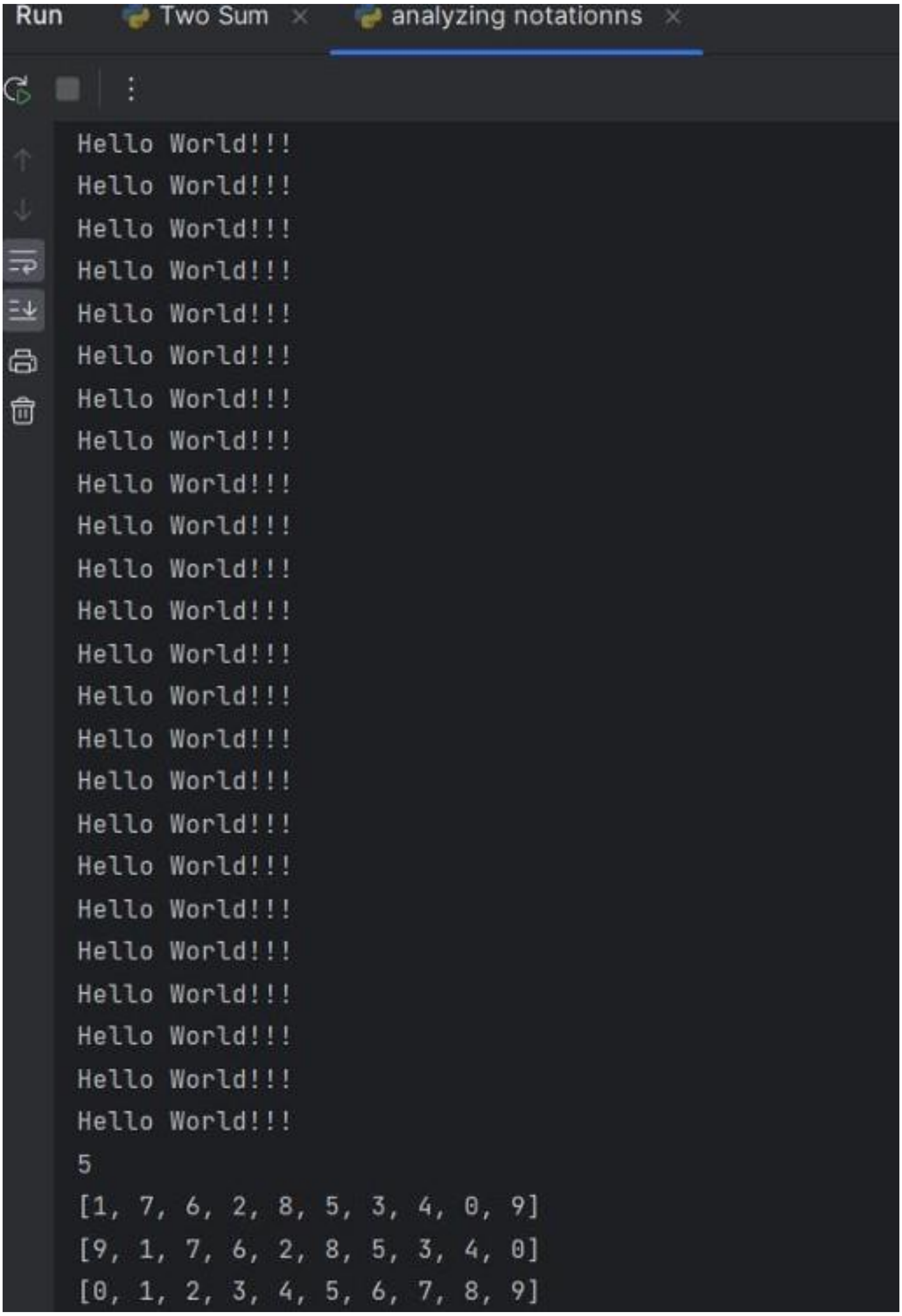


```
        swapped = False
        for i in range(len(data) - 1):
            if data[i] > data[i + 1]:
                data[i], data[i + 1] = data[i + 1], data[i]

# Example 7: Big Theta Notation (Θ)
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]
    return merge(merge_sort(left_half), merge_sort(right_half))

def merge(left, right):
    merged = []
    left_index = 0
    right_index = 0
    while left_index < len(left) and right_index < len(right):
        if left[left_index] <= right[right_index]:
```



```
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1
    merged.extend(left[left_index:])
    merged.extend(right[right_index:])
    return merged

# Example 8: Small O Notation (o)
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

n = 8
m = 5
example1(n)
example2(n)
example3(n, m)
example4(n)
print(fibonacci(5))
data = [9, 1, 7, 6, 2, 8, 5, 3, 4, 0]
bubble_sort(data)
print(data)
data = [9, 1, 7, 6, 2, 8, 5, 3, 4, 0]
merge_sort(data)
print(data)
data = [9, 1, 7, 6, 2, 8, 5, 3, 4, 0]
insertion_sort(data)
print(data)
```

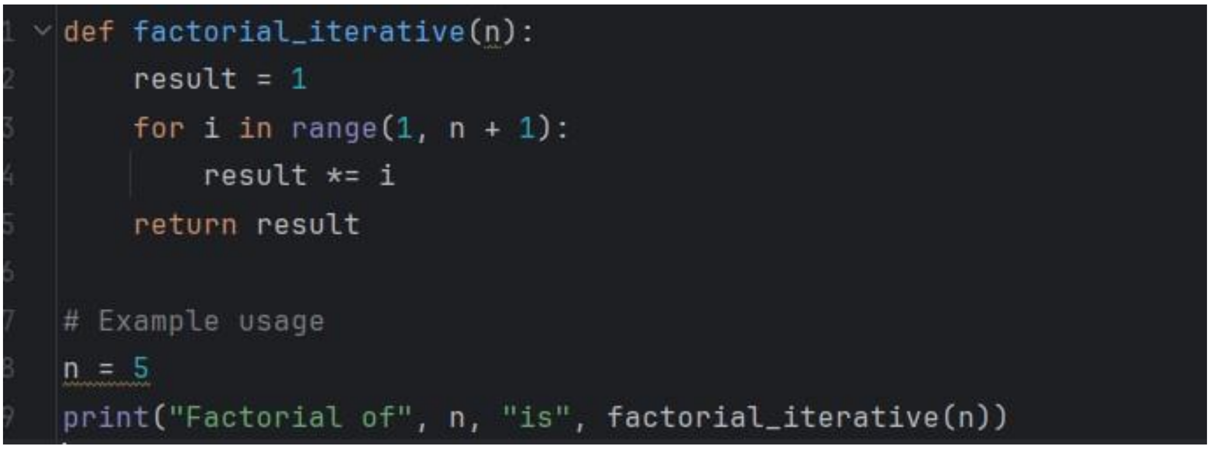
output



```
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
Hello World!!!
5
[1, 7, 6, 2, 8, 5, 3, 4, 0, 9]
[9, 1, 7, 6, 2, 8, 5, 3, 4, 0]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

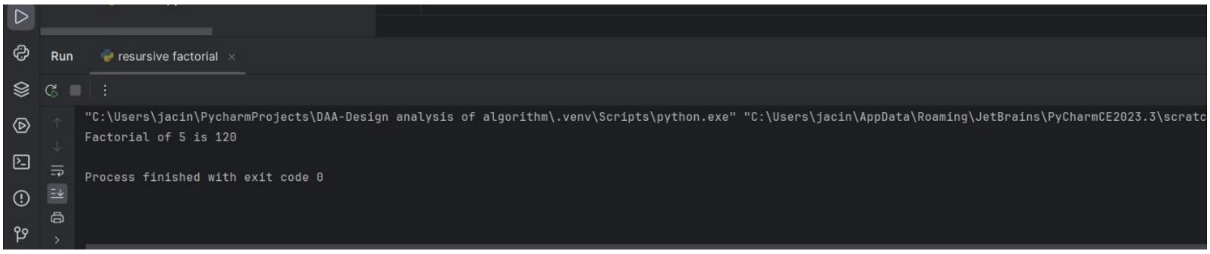
4.Write programs that demonstrate the mathematical analysis of non-recursive and recursive algorithms.

Coding



```python
def factorial_iterative(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

# Example usage
n = 5
print("Factorial of", n, "is", factorial_iterative(n))
```

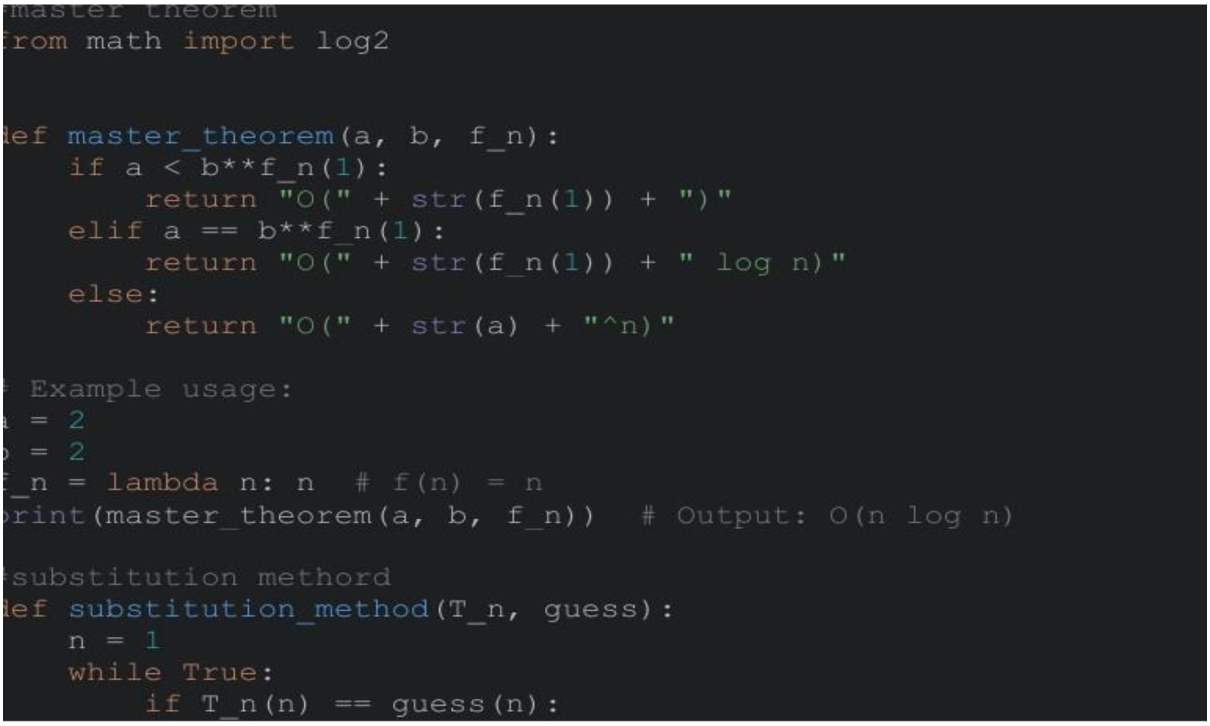
Output



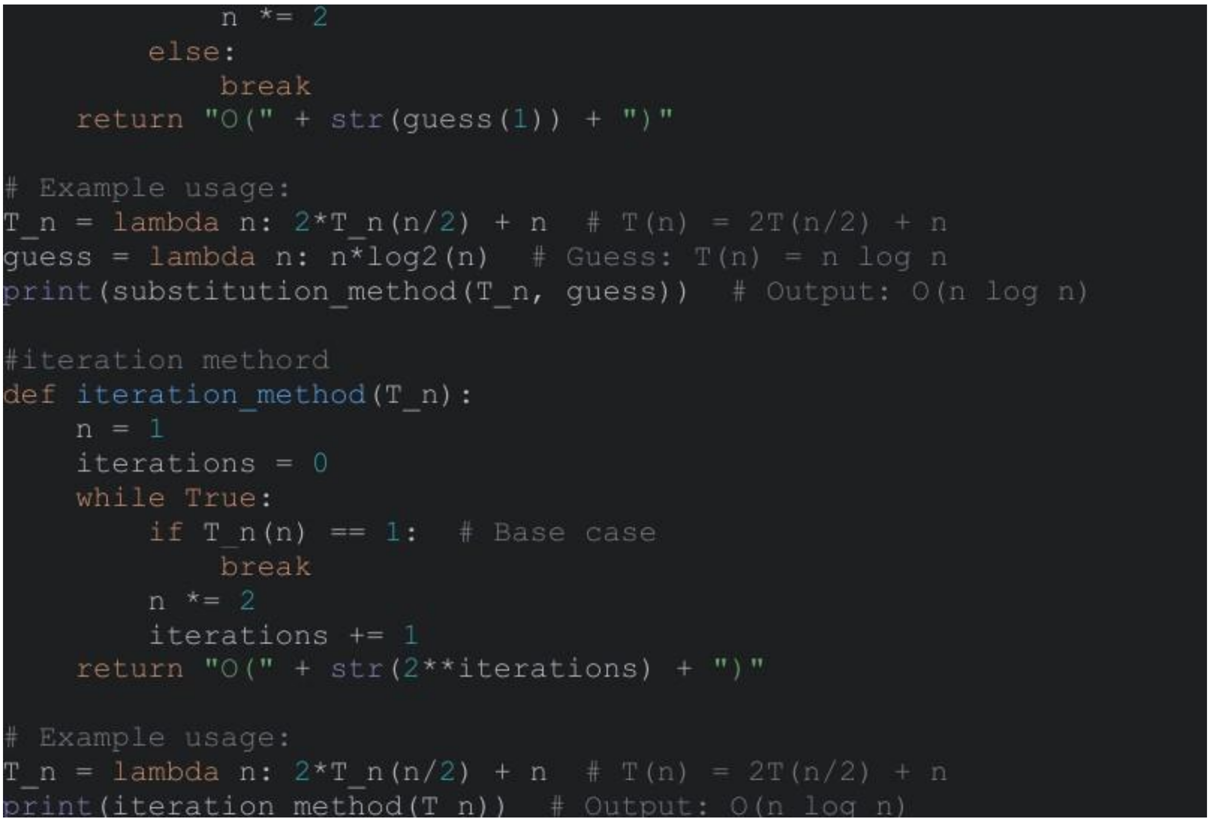
```
"C:\Users\jacin\PycharmProjects\DAA-Design analysis of algorithm\.venv\Scripts\python.exe" "C:\Users\jacin\AppData\Roaming\JetBrains\PyCharmCE2023.3\scratc
Factorial of 5 is 120

Process finished with exit code 0
```

5.Write programsforsolvingrecurrencerelationsusingtheMasterTheorem,Substitution Method,andIterationMethodwilldemonstratehowtocalculatethetimecomplexityof anexamplerecurrencerelationusingthespecifiedtechnique.

Coding



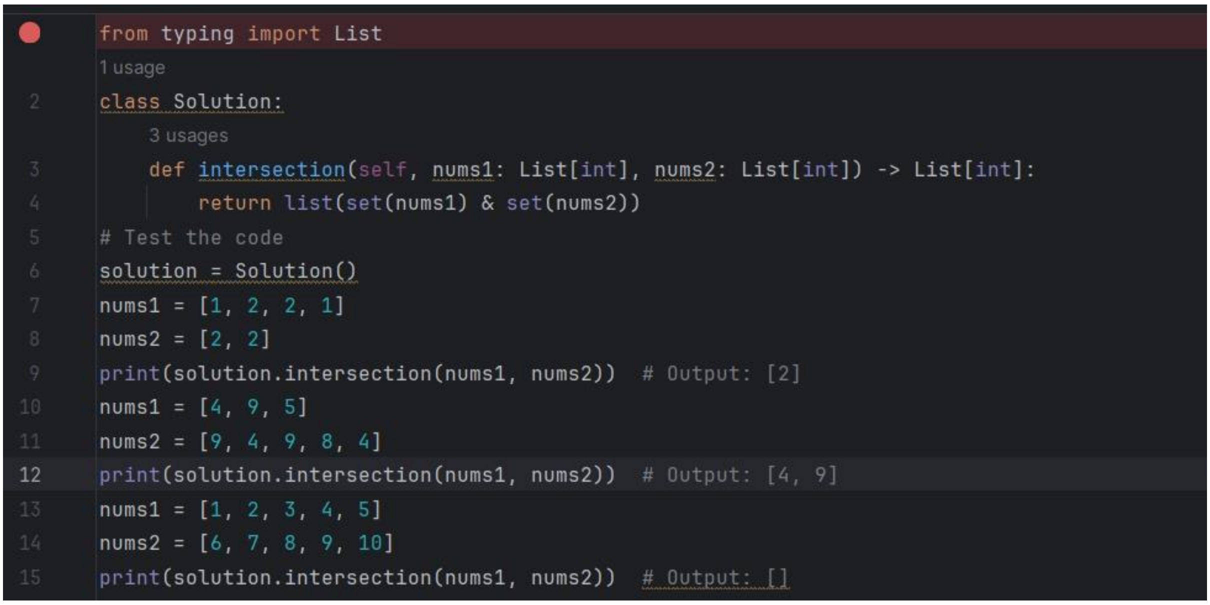
```python
#master theorem
from math import log2


def master_theorem(a, b, f_n):
    if a < b**f_n(1):
        return "O(" + str(f_n(1)) + ")"
    elif a == b**f_n(1):
        return "O(" + str(f_n(1)) + " log n)"
    else:
        return "O(" + str(a) + "^n)"

# Example usage:
a = 2
b = 2
f_n = lambda n: n  # f(n) = n
print(master_theorem(a, b, f_n))  # Output: O(n log n)

#substitution methord
def substitution_method(T_n, guess):
    n = 1
    while True:
        if T_n(n) == guess(n):
```



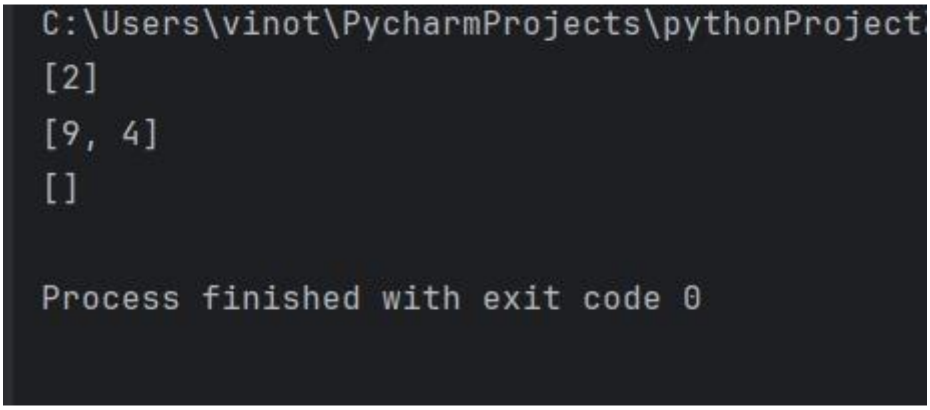
```
            n *= 2
        else:
            break
    return "O(" + str(guess(1)) + ")"

# Example usage:
T_n = lambda n: 2*T_n(n/2) + n  # T(n) = 2T(n/2) + n
guess = lambda n: n*log2(n)  # Guess: T(n) = n log n
print(substitution_method(T_n, guess))  # Output: O(n log n)

#iteration methord
def iteration_method(T_n):
    n = 1
    iterations = 0
    while True:
        if T_n(n) == 1:  # Base case
            break
        n *= 2
        iterations += 1
    return "O(" + str(2**iterations) + ")"

# Example usage:
T_n = lambda n: 2*T_n(n/2) + n  # T(n) = 2T(n/2) + n
print(iteration_method(T_n))  # Output: O(n log n)
```

6. Given two integer arrays nums1 and nums2, return an array of their Intersection. Each element in the result must be unique and you may return the result in any order.

Coding

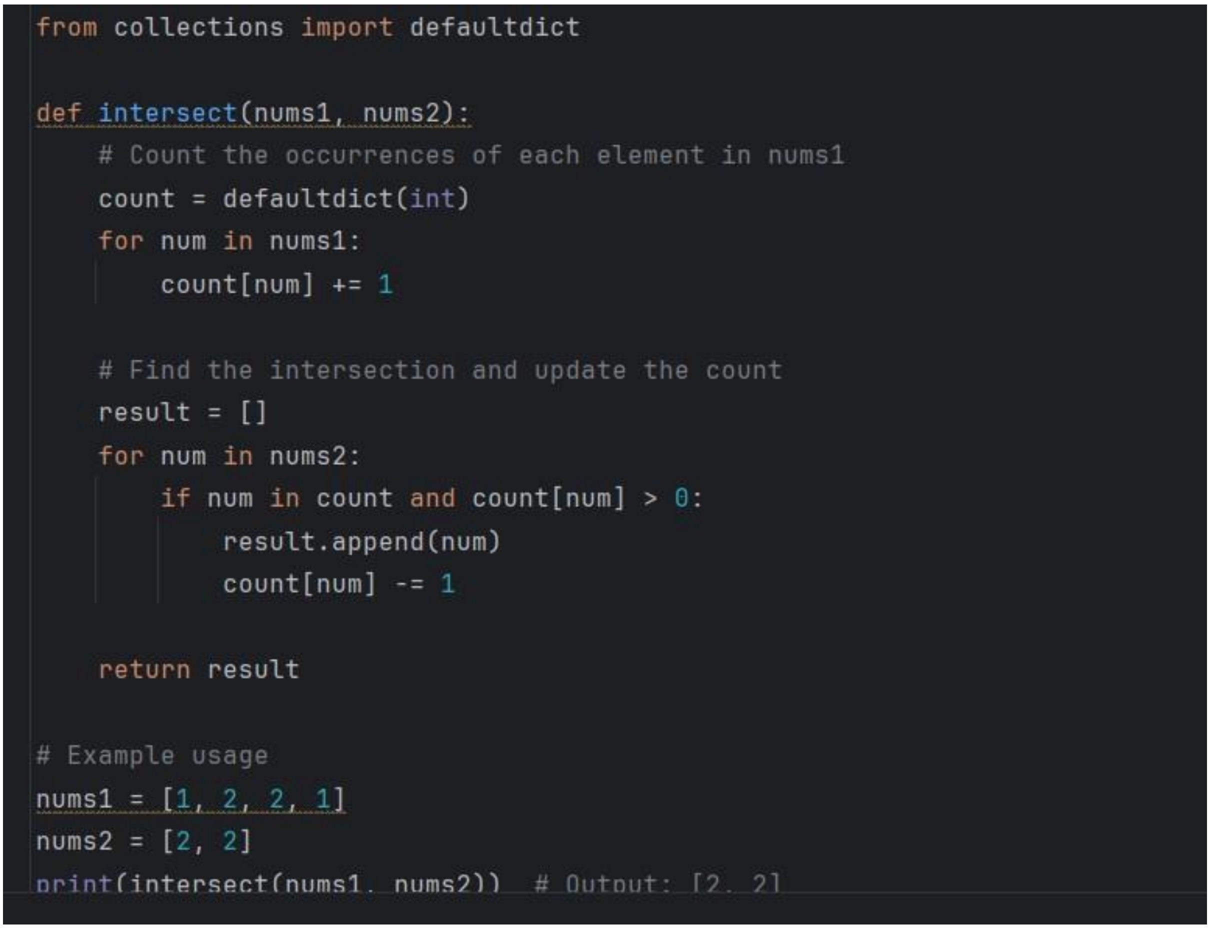


```
from typing import List
1 usage
class Solution:
    3 usages
    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:
        return list(set(nums1) & set(nums2))
# Test the code
solution = Solution()
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(solution.intersection(nums1, nums2))  # Output: [2]
nums1 = [4, 9, 5]
nums2 = [9, 4, 9, 8, 4]
print(solution.intersection(nums1, nums2))  # Output: [4, 9]
nums1 = [1, 2, 3, 4, 5]
nums2 = [6, 7, 8, 9, 10]
print(solution.intersection(nums1, nums2))  # Output: []
```

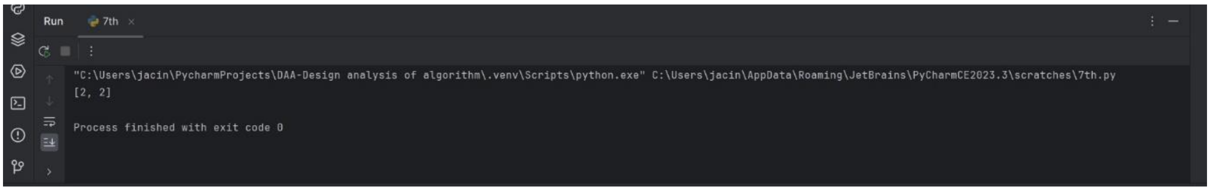


```
C:\Users\vinot\PycharmProjects\pythonProject
[2]
[9, 4]
[]

Process finished with exit code 0
```

7. Given two integer arrays nums1 and nums2, 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.

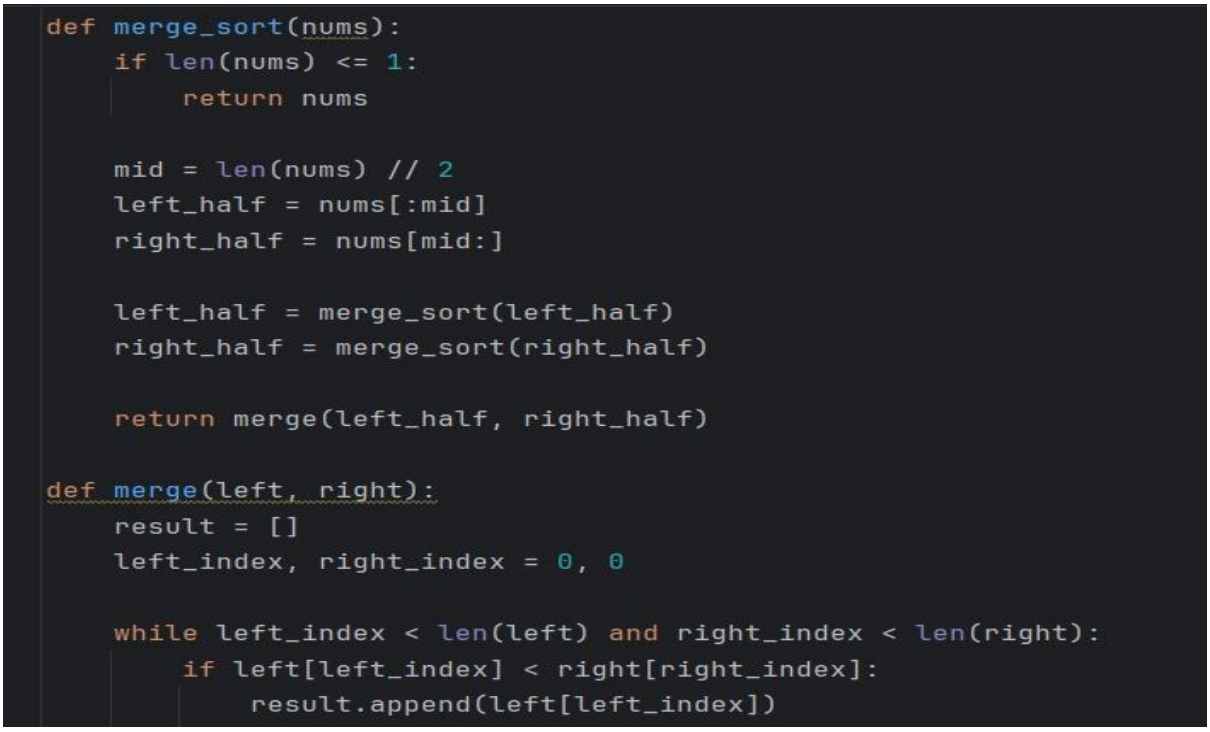


```python
from collections import defaultdict

def intersect(nums1, nums2):
    # Count the occurrences of each element in nums1
    count = defaultdict(int)
    for num in nums1:
        count[num] += 1

    # Find the intersection and update the count
    result = []
    for num in nums2:
        if num in count and count[num] > 0:
            result.append(num)
            count[num] -= 1

    return result

# Example usage
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersect(nums1, nums2))  # Output: [2, 2]
```

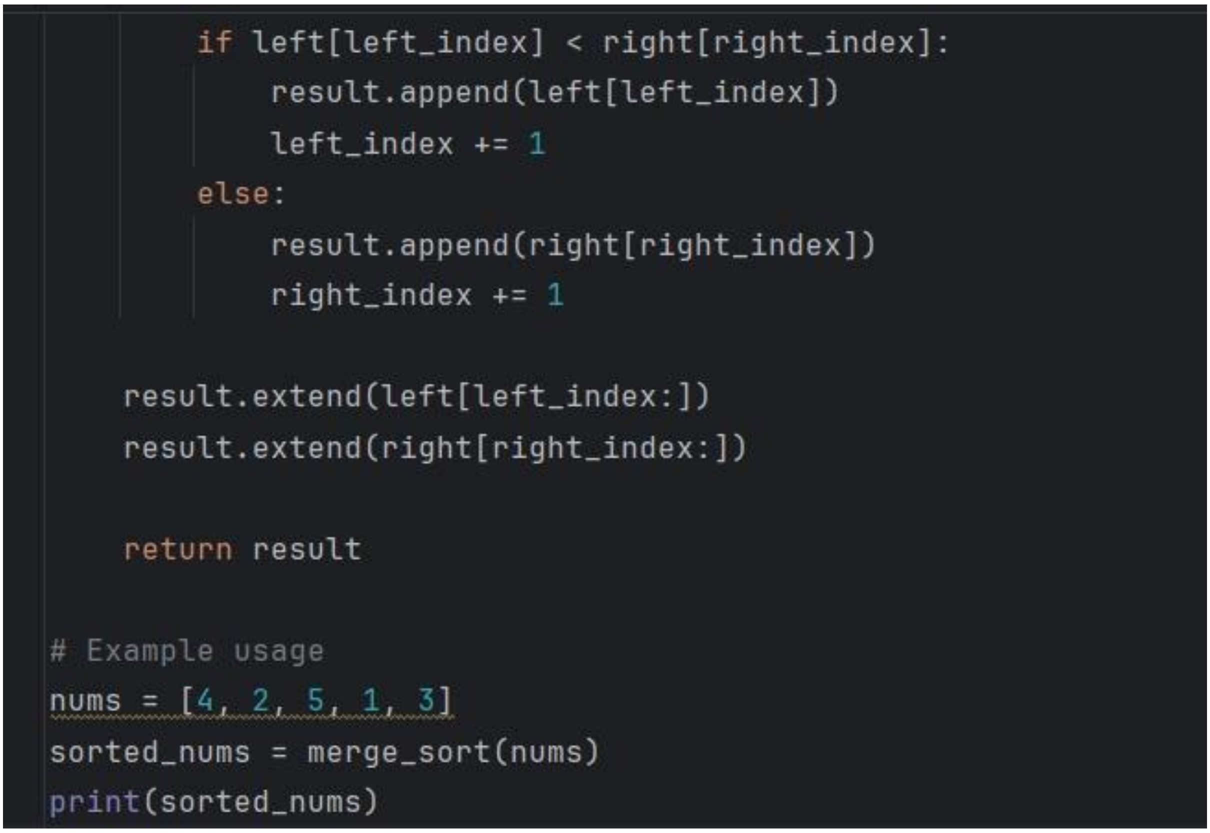


```
"C:\Users\jacin\PycharmProjects\DAA-Design analysis of algorithm\.venv\Scripts\python.exe" C:\Users\jacin\AppData\Roaming\JetBrains\PyCharmCE2023.3\scratches\7th.py
[2, 2]

Process finished with exit code 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

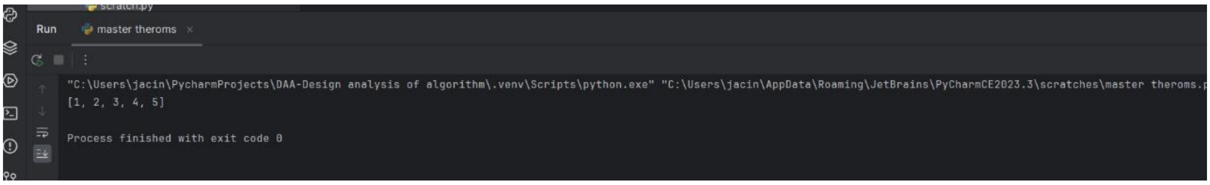


```python
def merge_sort(nums):
    if len(nums) <= 1:
        return nums

    mid = len(nums) // 2
    left_half = nums[:mid]
    right_half = nums[mid:]

    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    return merge(left_half, right_half)

def merge(left, right):
    result = []
    left_index, right_index = 0, 0

    while left_index < len(left) and right_index < len(right):
        if left[left_index] < right[right_index]:
            result.append(left[left_index])
```



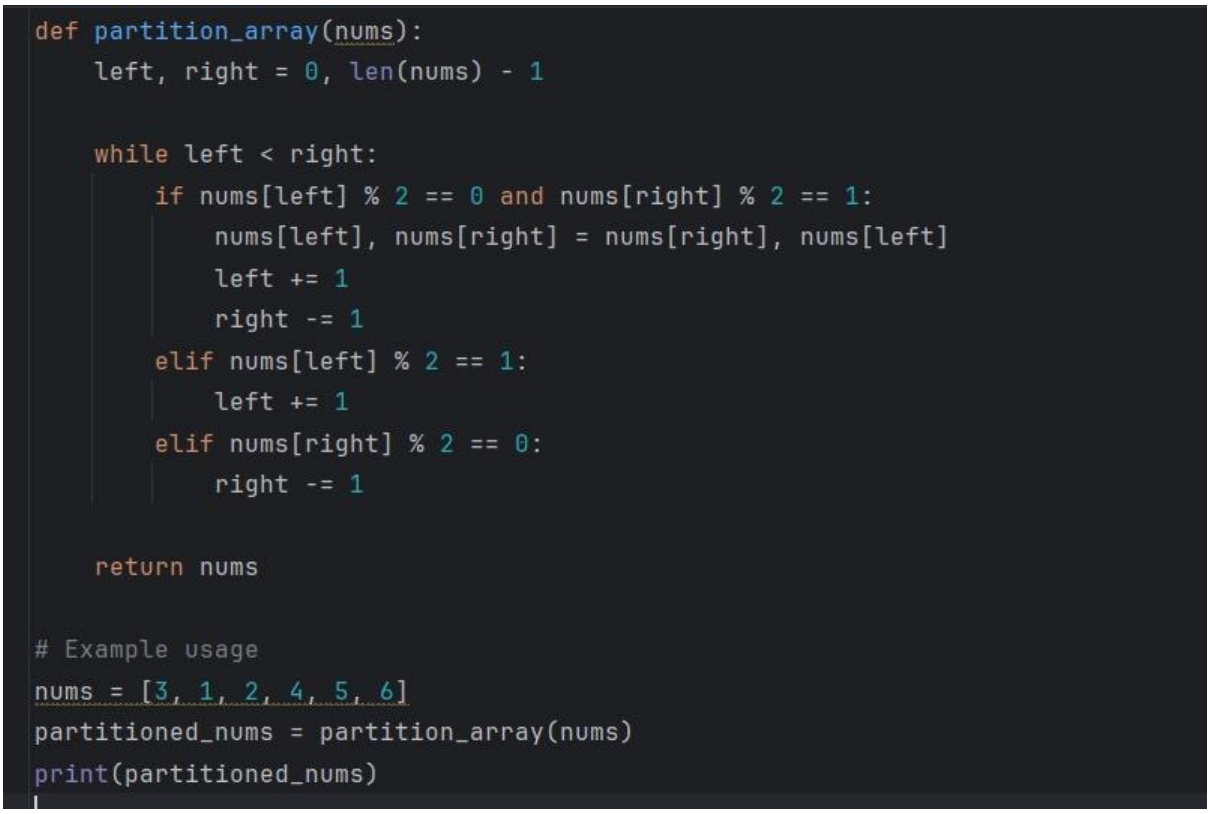
```python
        if left[left_index] < right[right_index]:
            result.append(left[left_index])
            left_index += 1
        else:
            result.append(right[right_index])
            right_index += 1

    result.extend(left[left_index:])
    result.extend(right[right_index:])

    return result

# Example usage
nums = [4, 2, 5, 1, 3]
sorted_nums = merge_sort(nums)
print(sorted_nums)
```

Output

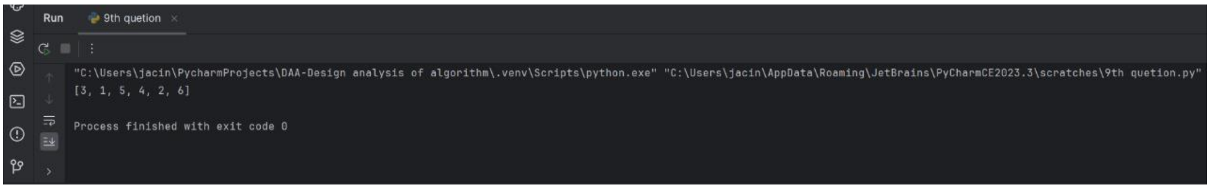


```
"C:\Users\jacin\PycharmProjects\DAA-Design analysis of algorithm\.venv\Scripts\python.exe" "C:\Users\jacin\AppData\Roaming\JetBrains\PyCharmCE2023.3\scratches\master theroms.p
[1, 2, 3, 4, 5]

Process finished with exit code 0
```

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



```python
def partition_array(nums):
    left, right = 0, len(nums) - 1

    while left < right:
        if nums[left] % 2 == 0 and nums[right] % 2 == 1:
            nums[left], nums[right] = nums[right], nums[left]
            left += 1
            right -= 1
        elif nums[left] % 2 == 1:
            left += 1
        elif nums[right] % 2 == 0:
            right -= 1

    return nums

# Example usage
nums = [3, 1, 2, 4, 5, 6]
partitioned_nums = partition_array(nums)
print(partitioned_nums)
```



```
"C:\Users\jacin\PycharmProjects\DAA-Design analysis of algorithm\.venv\Scripts\python.exe" "C:\Users\jacin\AppData\Roaming\JetBrains\PyCharmCE2023.3\scratches\9th quetion.py"
[3, 1, 5, 4, 2, 6]

Process finished with exit code 0
```