DATE: 4-6-24

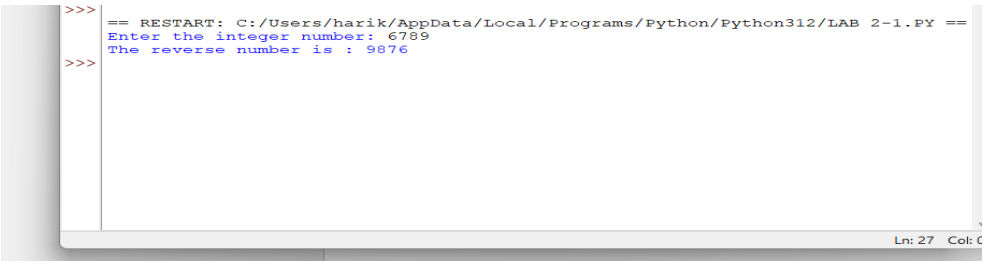
LAB-2

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

CODE:

```
# Initiate value to null
revs_number = 0

# reverse the integer number using the while loop

while (number > 0):
# Logic
remainder = number % 10
revs_number = (revs_number * 10) + remainder
number = number // 10

# Display the result
print("The reverse number is : {}".format(revs_number))
```

OUTPUT:



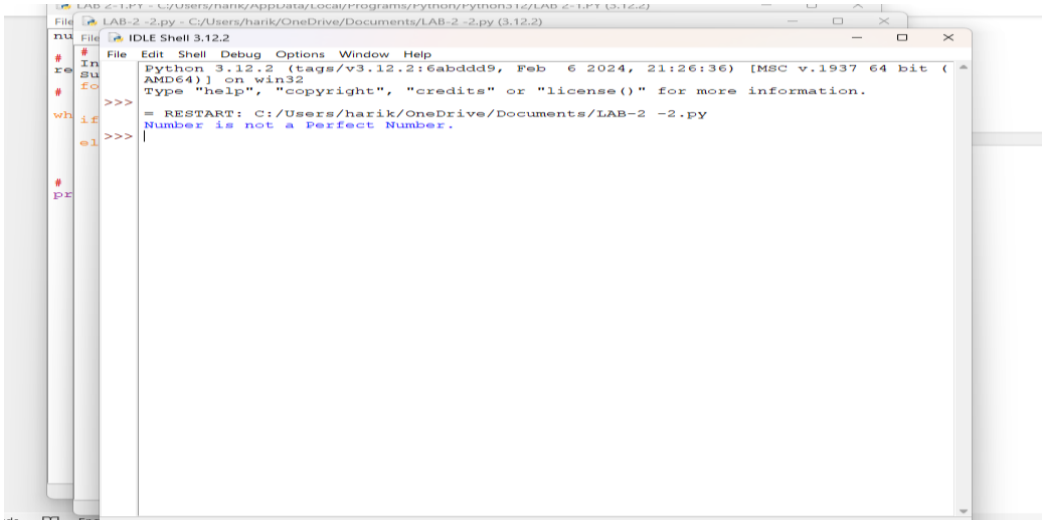
2.Write a program to find the perfect number

CODE:

```
# Perfect Number by using For_loop
Input_Number = 78
Sum = 0
for i in range(1, Input_Number):
    if(Input_Number % i == 0):
        Sum = Sum + i
```

```
    if (Sum == Input_Number):
        print("Number is a Perfect Number.")
    else:
        print("Number is not a Perfect Number.")
```

OUTPUT:

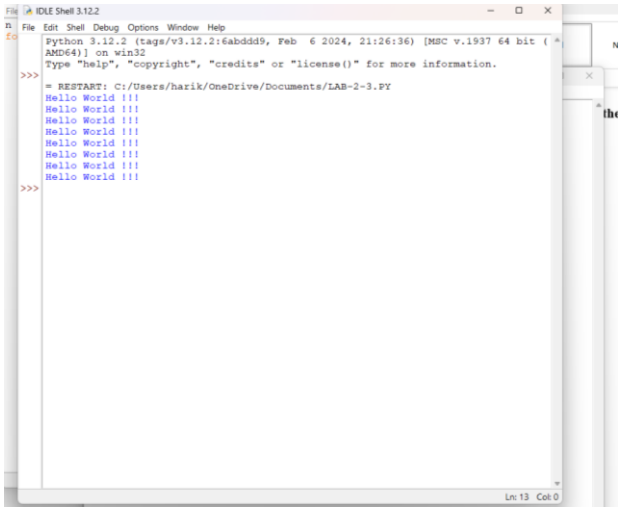


3. Write C program that demonstrates the usage of these notations by analyzing the time complexity of some example algorithms.

CODE:

```
n = 8
for i in range(1, n + 1):
    print("Hello World !!!")
```

OUTPUT:

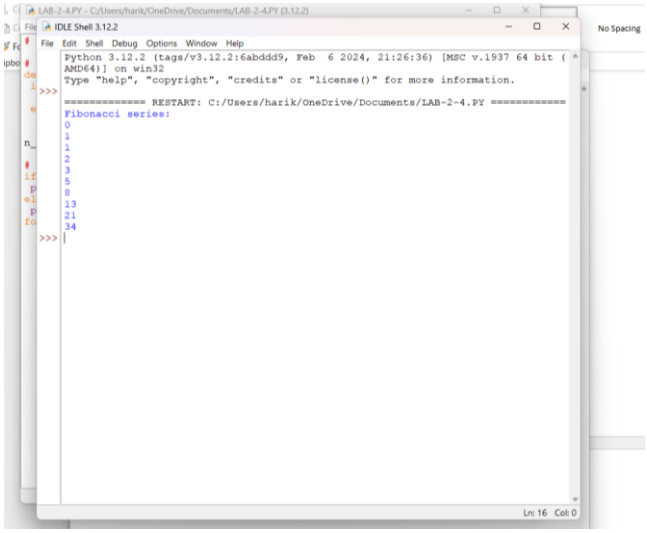


4. Write C programs that demonstrate the mathematical analysis of non-recursive and recursive algorithms.

CODE:

```
# Recursive function

def recursive_fibonacci(n):

if n <= 1:

return n

else:

return(recursive_fibonacci(n-1) + recursive_fibonacci(n-2))


n_terms = 10


# check if the number of terms is valid

if n_terms <= 0:

print("Invalid input ! Please input a positive value")

else:

print("Fibonacci series:")

for i in range(n_terms):

print(recursive_fibonacci(i))
```

OUTPUT:



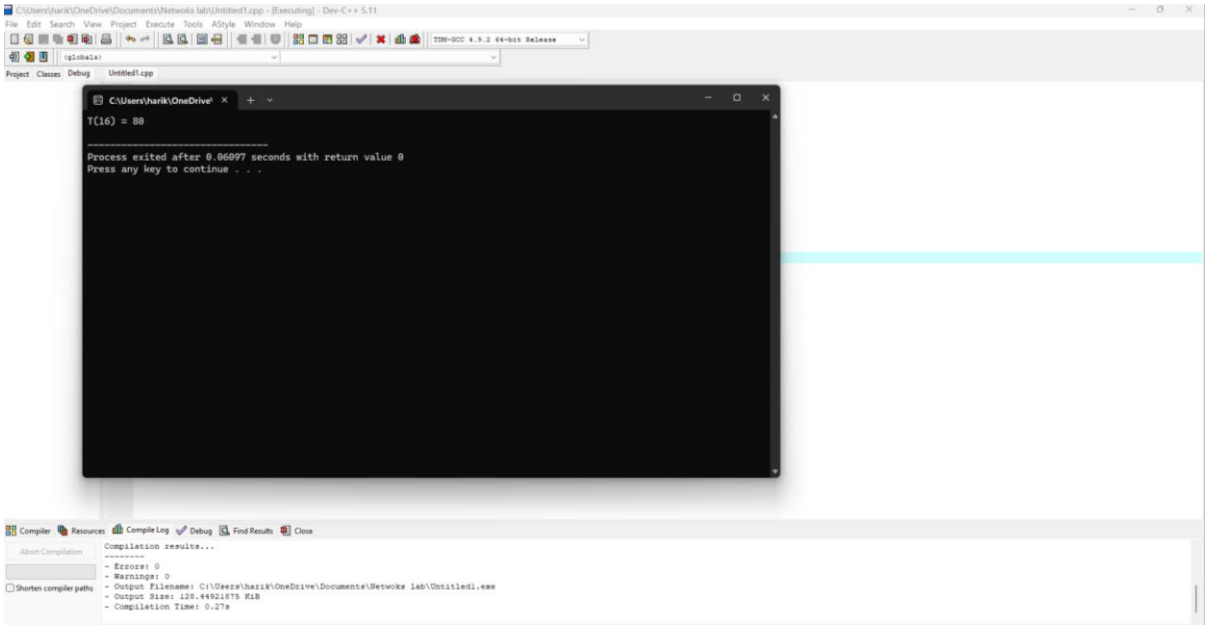
5. Write C programs for solving recurrence relations using the Master Theorem, Substitution Method, and Iteration Method will demonstrate how to calculate the time complexity of an example recurrence relation using the specified technique.

CODE:

```
#include <stdio.h>


// Function to implement T(n) = 2T(n/2) + n

int T(int n) {

    if (n == 1) {

        return 1;  // Base case

    } else {

        return 2 * T(n / 2) + n;

    }

}


int main() {

    int n = 16;  // Example input

    printf("T(%d) = %d\n", n, T(n));

    return 0;
```

}

OUTPUT:



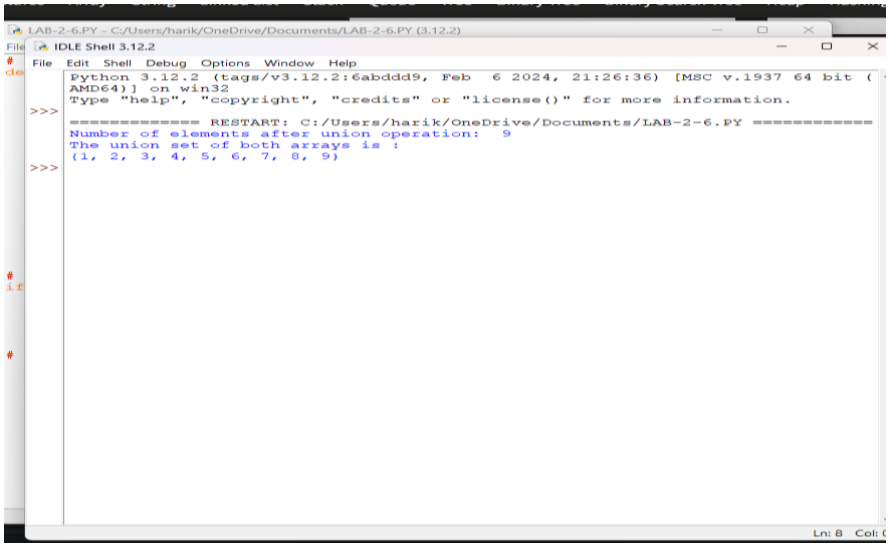
6. Given two integer arrays nums1 and nums2, return an array of their Intersection. Each element in the result must be unique and you may return the result in any order

CODE:

```
# Python program for the union of two arrays using Set

def getUnion(a, n, b, m):


        # Defining set container s

        s = set()


        # Inserting array elements in s

        for i in range(n):

                s.add(a[i])


        for i in range(m):

                s.add(b[i])
```

```
        print("Number of elements after union operation: ", len(s), "")
        print("The union set of both arrays is :" + "")

        print(s, end="") # s will contain only distinct
        # elements from array a and b


# Driver Code
if __name__ == '__main__':
        a = [1, 2, 5, 6, 2, 3, 5, 7, 3]
        b = [2, 4, 5, 6, 8, 9, 4, 6, 5, 4]

        getUnion(a, 9, b, 10)
```

OUTPUT:

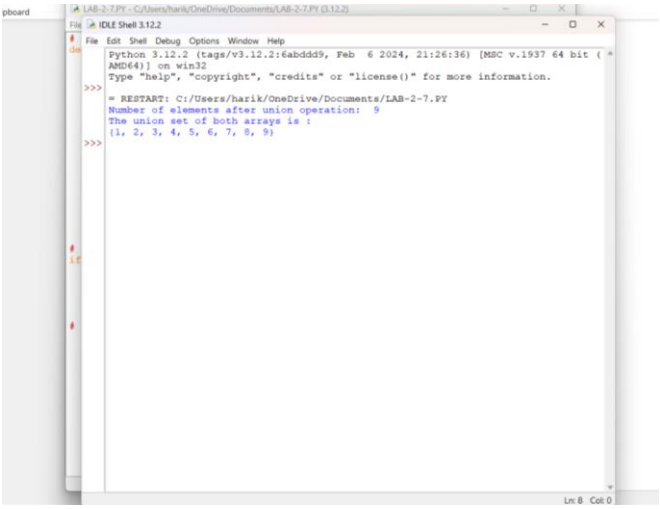


7. Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must appear as many times as it shows in both arrays and you may return the result in any order

CODE:

```
# Python program for the union of two arrays using Set
def getUnion(a, n, b, m):

    # Defining set container s
    s = set()

    # Inserting array elements in s
    for i in range(n):
        s.add(a[i])

    for i in range(m):
        s.add(b[i])
    print("Number of elements after union operation: ", len(s), "")
    print("The union set of both arrays is :" + "")

    print(s, end="") # s will contain only distinct
    # elements from array a and b



# Driver Code
if __name__ == '__main__':
    a = [1, 2, 5, 6, 2, 3, 5, 7, 3]
    b = [2, 4, 5, 6, 8, 9, 4, 6, 5, 4]

    getUnion(a, 9, b, 10)
```

OUTPUT:



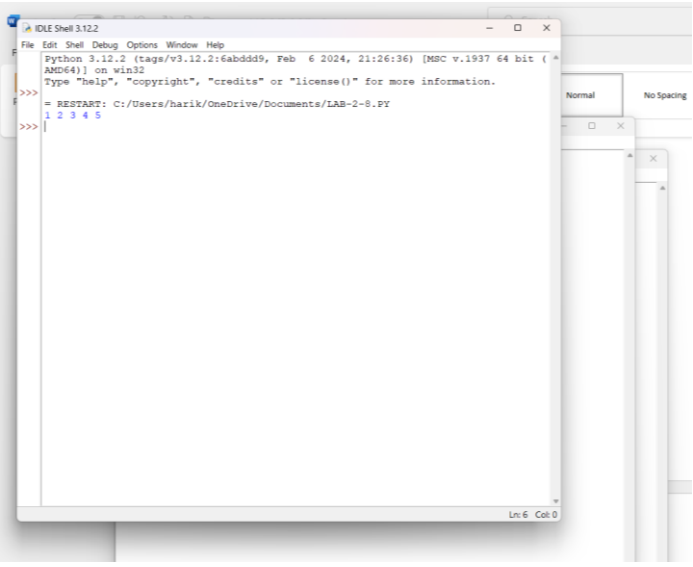
8. Given an array of integers nums, sort the array in ascending order and return it.You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

CODE:

```
# Python3 program for the above approach


# Function to print array element

def printArray(arr, N):


# Traverse the array

for i in range(N):

print(arr[i], end = ' ')


# Function to sort the array in O(N)

def sortArray(arr, N):


i = 0


# Traverse the array

while i < N:
```

```
# If the current element is
# at correct position
if arr[i] == i + 1:
i += 1

# Else swap the current element
# with it's correct position
else:

# Swap the value of
# arr[i] and arr[arr[i]-1]
temp1 = arr[i]
temp2 = arr[arr[i] - 1]
arr[i] = temp2
arr[temp1 - 1] = temp1

# Driver code
if __name__=='__main__':

arr = [ 2, 1, 5, 3, 4 ]
N = len(arr)

# Function call to sort the array
sortArray(arr, N)

# Function call to print the array
printArray(arr, N)
```

OUTPUT:



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

CODE:

```
# Python program to segregate even and odd elements of array


def segregateEvenOdd(arr):


# Initialize left and right indexes

left,right = 0,len(arr)-1


while left < right:


# Increment left index while we see 0 at left

while (arr[left]%2==0 and left < right):

left += 1


# Decrement right index while we see 1 at right

while (arr[right]%2 == 1 and left < right):

right -= 1
```

```
if (left < right):
# Swap arr[left] and arr[right]*/
arr[left],arr[right] = arr[right],arr[left]
left += 1
right = right-1



# Driver function to test above function
arr = [12, 34, 45, 9, 8, 90, 3]
segregateEvenOdd(arr)
print ("Array after segregation "),
for i in range(0,len(arr)):
print arr[i],
```

OUTPUT:

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.

CODE:

```
# Python program to segregate even and odd elements of array
```

```
def segregateEvenOdd(arr):

# Initialize left and right indexes
left,right = 0,len(arr)-1

while left < right:

# Increment left index while we see 0 at left
while (arr[left]%2==0 and left < right):
left += 1

# Decrement right index while we see 1 at right
while (arr[right]%2 == 1 and left < right):
right -= 1

if (left < right):
# Swap arr[left] and arr[right]*/
arr[left],arr[right] = arr[right],arr[left]
left += 1
right = right-1


# Driver function to test above function
arr = [12, 34, 45, 9, 8, 90, 3]
segregateEvenOdd(arr)
print ("Array after segregation "),
for i in range(0,len(arr)):
print arr[i],
```

OUTPUT:

```
>>>
========== RESTART: C:/Users/harik/OneDrive/Documents/LAB-2-10.py ==========
Array after segregation
12
34
90
8
9
45
3
>>>
```

Ln: 23 Col: 0