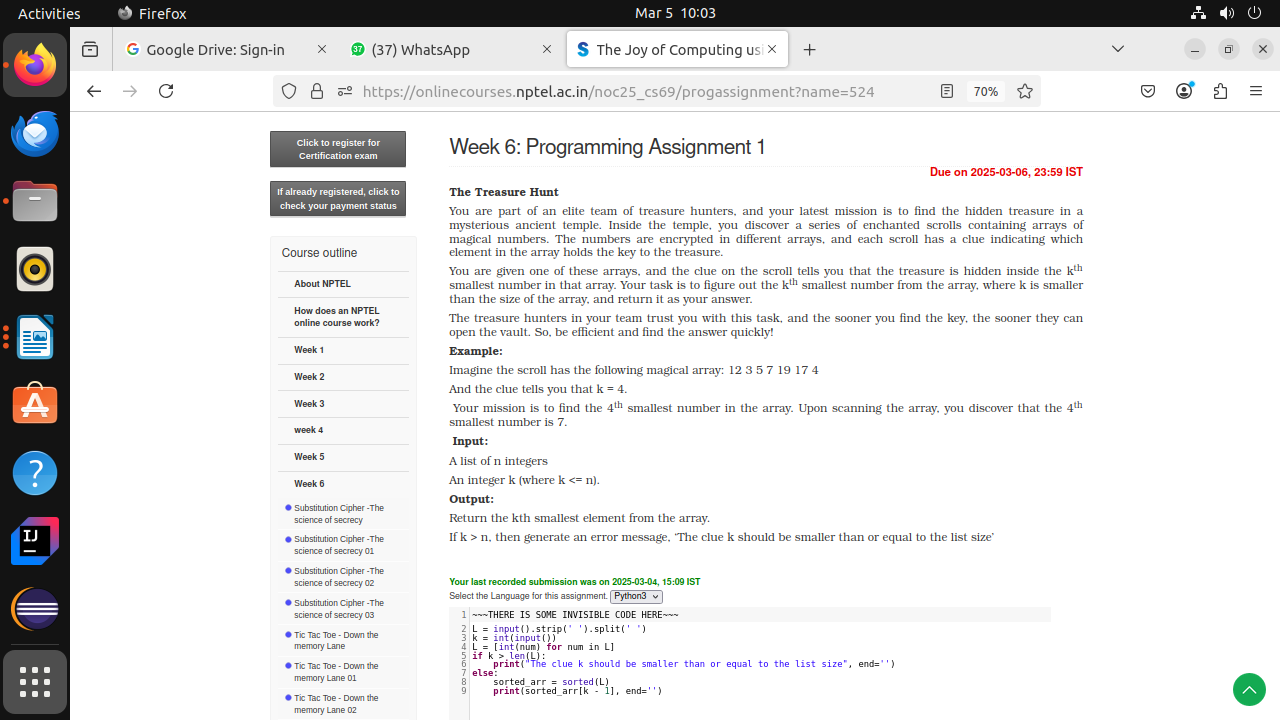
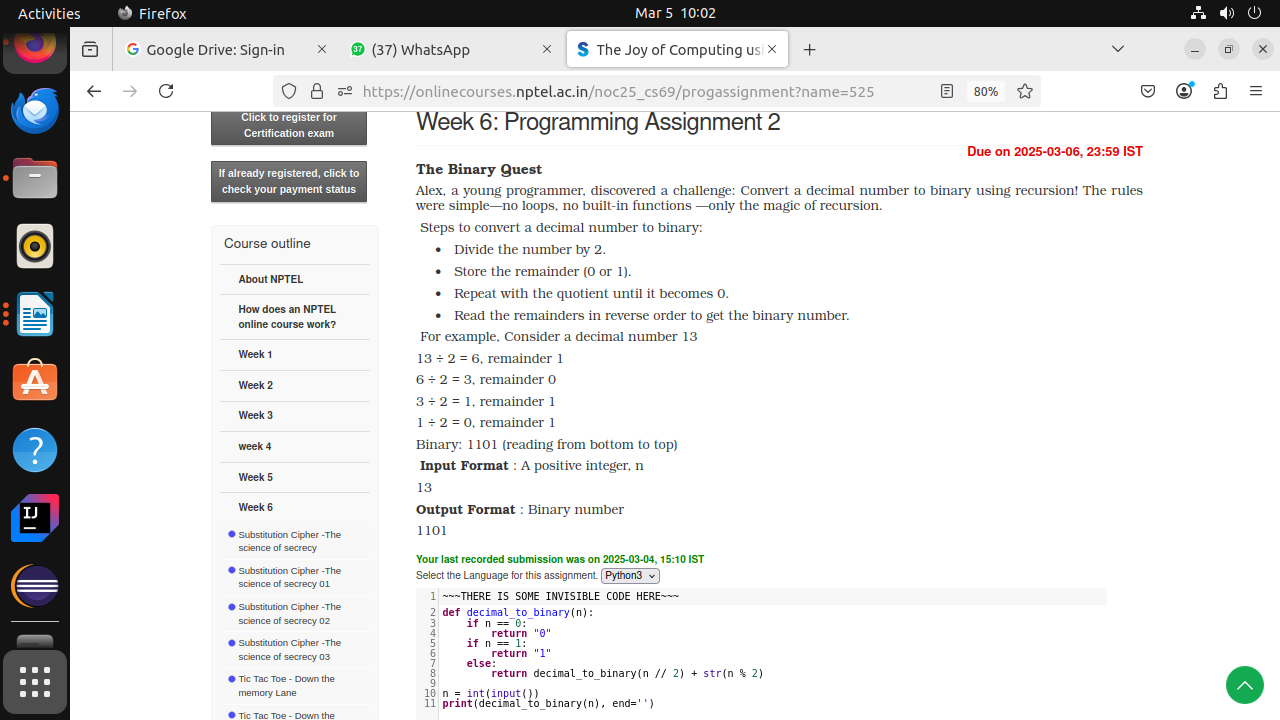
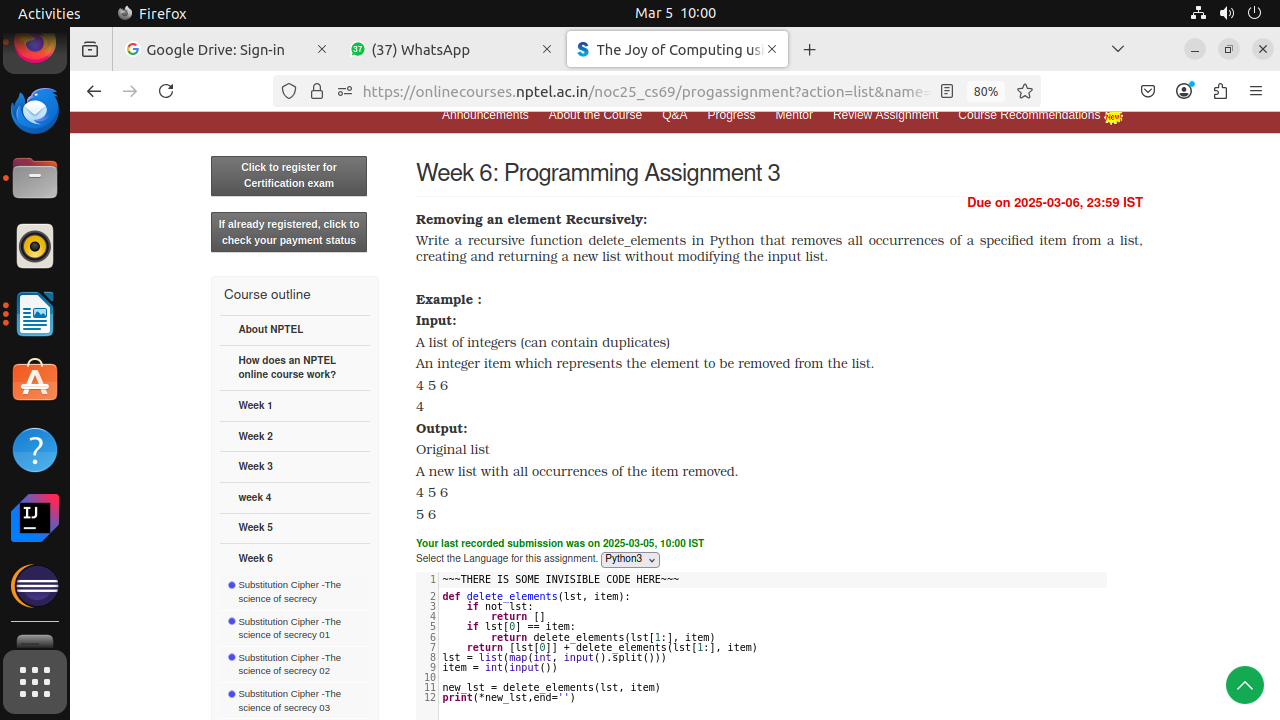
Assignment-1



Assignment-2



Assignment-3



**Assignment-1**

# **Week 6: Programming Assignment 1**

Due on 2025-03-06, 23:59 IST

**The Treasure Hunt**

You are part of an elite team of treasure hunters, and your latest mission is to find the hidden treasure in a mysterious ancient temple. Inside the temple, you discover a series of enchanted scrolls containing arrays of magical numbers. The numbers are encrypted in different arrays, and each scroll has a clue indicating which element in the array holds the key to the treasure.

You are given one of these arrays, and the clue on the scroll tells you that the treasure is hidden inside the kth smallest number in that array. Your task is to figure out the kth smallest number from the array, where k is smaller than the size of the array, and return it as your answer.

The treasure hunters in your team trust you with this task, and the sooner you find the key, the sooner they can open the vault. So, be efficient and find the answer quickly!

**Example:**

Imagine the scroll has the following magical array: 12 3 5 7 19 17 4

And the clue tells you that k = 4.

Your mission is to find the 4th smallest number in the array. Upon scanning the array, you discover that the 4th smallest number is 7.

**Input:**

A list of n integers

An integer k (where k <= n).

**Output:**

Return the kth smallest element from the array.

If k > n, then generate an error message, ‘The clue k should be smaller than or equal to the list size’

L = input().strip(' ').split(' ') # Read input, remove leading/trailing spaces, and split by spaces

k = int(input()) # Read the second input as an integer (k)

L = [int(num) for num in L] # Convert list elements from string to integer

if k > len(L): # Check if k is larger than the list size

print("The clue k should be smaller than or equal to the list size", end='')

else:

sorted\_arr = sorted(L) # Sort the list in ascending order

print(sorted\_arr[k - 1], end='') # Print the k-th smallest element (1-based index)

**Assignment-2**

# **Week 6: Programming Assignment 2**

Due on 2025-03-06, 23:59 IST

**The Binary Quest**

Alex, a young programmer, discovered a challenge: Convert a decimal number to binary using recursion! The rules were simple—no loops, no built-in functions —only the magic of recursion.

 Steps to convert a decimal number to binary:

·Divide the number by 2.

·Store the remainder (0 or 1).

·Repeat with the quotient until it becomes 0.

·Read the remainders in reverse order to get the binary number.

 For example, Consider a decimal number 13

13 ÷ 2 = 6, remainder 1

6 ÷ 2 = 3, remainder 0

3 ÷ 2 = 1, remainder 1

1 ÷ 2 = 0, remainder 1

Binary: 1101 (reading from bottom to top)

**Input Format** : A positive integer, n

13

**Output Format** : Binary number

1101

# Function to convert decimal number to binary using recursion

def decimal\_to\_binary(n):

# Base Case 1: If n is 0, return "0"

if n == 0:

return "0"

# Base Case 2: If n is 1, return "1"

if n == 1:

return "1"

# Recursive Case: Divide the number by 2 and append the remainder

# Explanation:

# - decimal\_to\_binary(n // 2) calls the function for the quotient

# - str(n % 2) gets the remainder (0 or 1) as a string

return decimal\_to\_binary(n // 2) + str(n % 2)

# Read an integer from user input

n = int(input("Enter a decimal number: "))

# Convert decimal to binary and print the result

print("Binary representation:", decimal\_to\_binary(n), end='')

**Assignment-3**

# **Week 6: Programming Assignment 3**

Due on 2025-03-06, 23:59 IST

**Removing an element Recursively:**

Write a recursive function delete\_elements in Python that removes all occurrences of a specified item from a list, creating and returning a new list without modifying the input list.

**Example :**

**Input:**

A list of integers (can contain duplicates)

An integer item which represents the element to be removed from the list.

4 5 6

4

**Output:**

Original list

A new list with all occurrences of the item removed.

4 5 6

5 6

def delete\_elements(lst, item):

if not lst: # Base case: if the list is empty, return an empty list

return []

if lst[0] == item: # If the first element matches the item, skip it

return delete\_elements(lst[1:], item)

return [lst[0]] + delete\_elements(lst[1:], item) # Include the first element and recurse

# Taking input dynamically

lst = list(map(int, input().split())) # Read space-separated integers

item = int(input()) # Read the item to remove

new\_lst = delete\_elements(lst, item)

# Printing expected output format

print(\*new\_lst,end='') # Print list elements space-separated

def delete\_elements(lst, item):

if not lst: # Base case: if the list is empty, return an empty list

return []

if lst[0] == item: # If the first element matches the item, skip it

return delete\_elements(lst[1:], item)

return [lst[0]] + delete\_elements(lst[1:], item) # Include the first element and recurse

# Taking input dynamically

lst = list(map(int, input().split())) # Read space-separated integers

item = int(input()) # Read the item to remove

new\_lst = delete\_elements(lst, item)

# Printing expected output format

print(\*new\_lst,end='') # Print list elements space-separated

Assignment-1

L = input().strip(' ').split(' ')

k = int(input())

L = [int(num) for num in L]

if k > len(L):

print("The clue k should be smaller than or equal to the list size", end='')

else:

sorted\_arr = sorted(L)

print(sorted\_arr[k - 1], end='')

Assignment-2

def decimal\_to\_binary(n):

if n == 0:

return "0"

if n == 1:

return "1"

else:

return decimal\_to\_binary(n // 2) + str(n % 2)

n = int(input())

print(decimal\_to\_binary(n), end='')

Assignment-3

def delete\_elements(lst, item):

if not lst:

return []

if lst[0] == item:

return delete\_elements(lst[1:], item)

return [lst[0]] + delete\_elements(lst[1:], item)

lst = list(map(int, input().split()))

item = int(input())

new\_lst = delete\_elements(lst, item)

print(\*new\_lst,end='')