

## ASSIGNMENT 2:

### 1 . Container With Most Water

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container

**CODE :**

```
def maxArea(A, Len) :
    area = 0
    for i in range(Len) :
        for j in range(i + 1, Len) :

            # Calculating the max area
            area = max(area, min(A[j], A[i]) * (j - i))

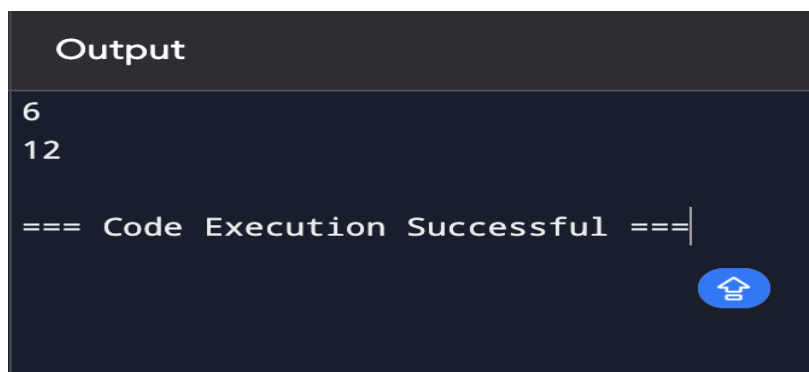
    return area

# Driver code
a = [ 1, 5, 4, 3 ]
b = [ 3, 1, 2, 4, 5 ]

len1 = len(a)
print(maxArea(a, len1))

len2 = len(b)
print(maxArea(b, len2))
```

**OUTPUT :**



### 2 . Integer to Roman

**CODE :**

```
def printRoman(number):
    num = [1, 4, 5, 9, 10, 40, 50, 90,
           100, 400, 500, 900, 1000]
    sym = ["I", "IV", "V", "IX", "X", "XL",
           "L", "XC", "C", "CD", "D", "CM", "M"]
    i = 12
```

## ASSIGNMENT 2:

```
while number:
    div = number // num[i]
    number %= num[i]

    while div:
        print(sym[i], end = "")
        div -= 1
    i -= 1

# Driver code
if __name__ == "__main__":
    number = 3549
    print("Roman value is:", end = " ")
    printRoman(number)
```

**OUTPUT :**

### Output

```
Roman value is: MMMDXLIX
=== Code Execution Successful ===
```

### 3 . Roman to Integer

**CODE :**

```
def value(r):
    if (r == 'I'):
        return 1
    if (r == 'V'):
        return 5
    if (r == 'X'):
        return 10
    if (r == 'L'):
        return 50
    if (r == 'C'):
        return 100
    if (r == 'D'):
        return 500
    if (r == 'M'):
```

## ASSIGNMENT 2:

```
        return 1000
    return -1
def romanToDecimal(str):
    res = 0
    i = 0
    while (i < len(str)):
        s1 = value(str[i])

        if (i + 1 < len(str)):
            s2 = value(str[i + 1])
            if (s1 >= s2):
                res = res + s1
                i = i + 1
            else:
                res = res + s2 - s1
                i = i + 2
        else:
            res = res + s1
            i = i + 1
    return res
# Driver code
print("Integer form of Roman Numeral is"),
print(romanToDecimal("MCMIV"))
```

**OUTPUT :**

```
Output
Integer form of Roman Numeral is
1904

=== Code Execution Successful ===
```

### 4 . Longest Common Prefix

**CODE :**

```
def longestCommonPrefix( a):
    size = len(a)
    if (size == 0):
        return ""

    if (size == 1):
```

## ASSIGNMENT 2:

```
        return a[0]
    a.sort()
    end = min(len(a[0]), len(a[size - 1]))
    i = 0
    while (i < end and
           a[0][i] == a[size - 1][i]):
        i += 1

    pre = a[0][0: i]
    return pre
```

# Driver Code

```
if __name__ == "__main__":
```

```
    input = ["geeksforgeeks", "geeks",
              "geek", "geezer"]
    print("The longest Common Prefix is :",
          longestCommonPrefix(input))
```

**OUTPUT :**

**Output**

The longest Common Prefix is : gee

=== Code Execution Successful ===

### 5 . 3Sum

**CODE :**

```
class Solution(object):
```

```
    def threeSum(self, nums):
```

```
        nums.sort()
```

```
        result = []
```

```
        for i in range(len(nums)-2):
```

```
            if i > 0 and nums[i] == nums[i-1]:
```

```
                continue
```

```
            l = i+1
```

## ASSIGNMENT 2:

```
r = len(nums)-1
while(l<r):
    sum = nums[i] + nums[l] + nums[r]
    if sum<0:
        l+=1
    elif sum >0:
        r-=1
    else:
        result.append([nums[i],nums[l],nums[r]])
        while l<len(nums)-1 and nums[l] == nums[l + 1] : l += 1
        while r>0 and nums[r] == nums[r - 1]: r -= 1
        l+=1
        r-=1
return result
ob1 = Solution()
print(ob1.threeSum([-1,0,1,2,-1,-4]))
```

**OUTPUT :**

```
Output
[[-1, -1, 2], [-1, 0, 1]]

=== Code Execution Successful ===
```

### **6 . 3Sum Closest**

**CODE :**

```
import sys
def solution(arr, x):
    closestSum = sys.maxsize
    for i in range (len(arr)) :
        for j in range(i + 1, len(arr)):
            for k in range(j + 1, len( arr)):
                if(abs(x - closestSum) >
```

## ASSIGNMENT 2:

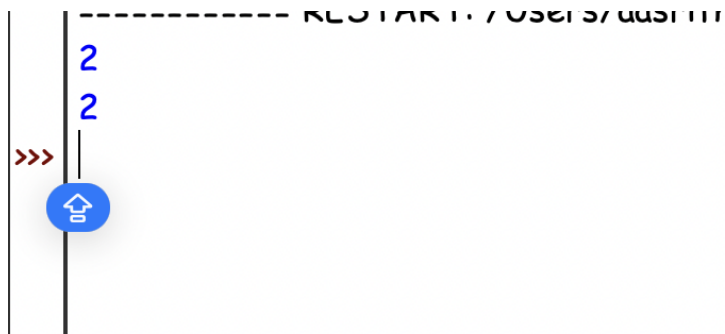
```
        abs(x - (arr[i] +
arr[j] + arr[k]))):
        closestSum = (arr[i] +arr[j] + arr[k])

return closestSum
```

# Driver code

```
if __name__ == "__main__":
    arr = [ -1, 2, 1, -4 ]
    x = 1
    print(solution(arr, x))
```

OUTPUT :



### 7 . Letter Combinations of a Phone Number

CODE :

```
def letterCombinationsUtil(number, n, table):
    list = []
    q = deque()
    q.append("")

    while len(q) != 0:
        s = q.pop()
        if len(s) == n:
            list.append(s)
        else:
            for letter in table[number[len(s)]]:
                q.append(s + letter)

    return list
```

## ASSIGNMENT 2:

```
def letterCombinations(number, n):
```

```
    # table[i] stores all characters that
    # corresponds to ith digit in phone
    table = ["0", "1", "abc", "def", "ghi", "jkl",
             "mno", "pqrs", "tuv", "wxyz"]
```

```
    list = letterCombinationsUtil(number, n, table)
```

```
    s = ""
```

```
    for word in list:
```

```
        s += word + " "
```

```
    print(s)
```

```
    return
```

```
# Driver code
```

```
number = [2, 3]
```

```
n = len(number)
```

```
letterCombinations(number, n)
```

OUTPUT :

### Output

```
cf ce cd bf be bd af ae ad
```

```
=== Code Execution Successful ===
```



### 8 . 4Sum

CODE :

```
class pairSum:
```

```
    def __init__(self):
```

```
        self.first = ""
```

## ASSIGNMENT 2:

```
        self.sec = ""
        self.sum = ""
def noCommon(a, b):
    if (a.first == b.first or a.first == b.sec or a.sec == b.first or a.sec == b.sec):
        return False
    return True
def findFourElements(myArr, sum):
    length = len(myArr)
    size = ((length * (length - 1)) // 2)
    aux = [None for _ in range(size)]
    k = 0
    for i in range(length - 1):
        for j in range(i + 1, length):
            aux[k] = pairSum()
            aux[k].sum = myArr[i] + myArr[j]
            aux[k].first = i
            aux[k].sec = j
            k += 1
    aux.sort(key=lambda x: x.sum)
    i = 0
    j = size - 1
    while (i < size and j >= 0):
        if ((aux[i].sum + aux[j].sum == sum)
            and noCommon(aux[i], aux[j])):
            print(myArr[aux[i].first], myArr[aux[i].sec],
                  myArr[aux[j].first], myArr[aux[j].sec], sep=", ")
            return

        elif (aux[i].sum + aux[j].sum < sum):
            i += 1
        else:
            j -= 1
```



## ASSIGNMENT 2:

# Driver Code

```
arr = [10, 20, 30, 40, 1, 2]
```

```
X = 91
```

```
findFourElements(arr, X)
```

**OUTPUT :**

**Output**

20, 1, 30, 40

=== Code Execution Successful ===

### 9 . Remove Nth Node From End of List

**CODE :**

```
class Node:
```

```
    def __init__(self, value):
```

```
        self.data = value
```

```
        self.next = None
```

```
def length(head):
```

```
    temp = head
```

```
    count = 0
```

```
    while(temp != None):
```

```
        count += 1
```

```
        temp = temp.next
```

```
    return count
```

```
def printList(head):
```

```
    ptr = head
```

```
    while(ptr != None):
```

```
        print (ptr.data, end = " ")
```

```
        ptr = ptr.next
```

## **ASSIGNMENT 2:**

```
print()
```

```
def deleteNthNodeFromEnd(head, n):  
    Length = length(head)  
    nodeFromBeginning = Length - n + 1  
    prev = None  
    temp = head  
    for i in range(1, nodeFromBeginning):  
        prev = temp  
        temp = temp.next  
    if(prev == None):  
        head = head.next  
        return head  
    else:  
        prev.next = prev.next.next  
        return head
```

```
if __name__ == '__main__':  
    head = Node(1)  
    head.next = Node(2)  
    head.next.next = Node(3)  
    head.next.next.next = Node(4)  
    head.next.next.next.next = Node(5)  
    print("Linked List before Deletion:")  
    printList(head)  
  
    head = deleteNthNodeFromEnd(head, 4)  
  
    print("Linked List after Deletion:")  
    printList(head)
```

## ASSIGNMENT 2:

OUTPUT :

### Output

Linked List before Deletion:

1 2 3 4 5

Linked List after Deletion:

1 3 4 5

### 10 . Valid Parentheses

```
open_list = ["[","{","("]
```

```
close_list = ["]","}",")"]
```

```
def check(myStr):
```

```
    stack = []
```

```
    for i in myStr:
```

```
        if i in open_list:
```

```
            stack.append(i)
```

```
        elif i in close_list:
```

```
            pos = close_list.index(i)
```

```
            if ((len(stack) > 0) and
```

```
                (open_list[pos] == stack[len(stack)-1]]):
```

```
                stack.pop()
```

```
            else:
```

```
                return "Unbalanced"
```

```
    if len(stack) == 0:
```

```
        return "Balanced"
```

```
    else:
```

```
        return "Unbalanced"
```

```
# Driver code
```

```
string = "[[]{()}"
```

```
print(string,"-", check(string))
```

## ASSIGNMENT 2:

```
string = "[{}]()"  
print(string,"-", check(string))  
string = "(()"  
print(string,"-",check(string))
```

OUTPUT :

### Output

```
{[]{()}} - Balanced  
[{}{ })() - Unbalanced  
((() - Unbalanced
```

```
=== Code Execution Successful ===
```