LEETCODE PROGRAMS

Dt:07-06-2024

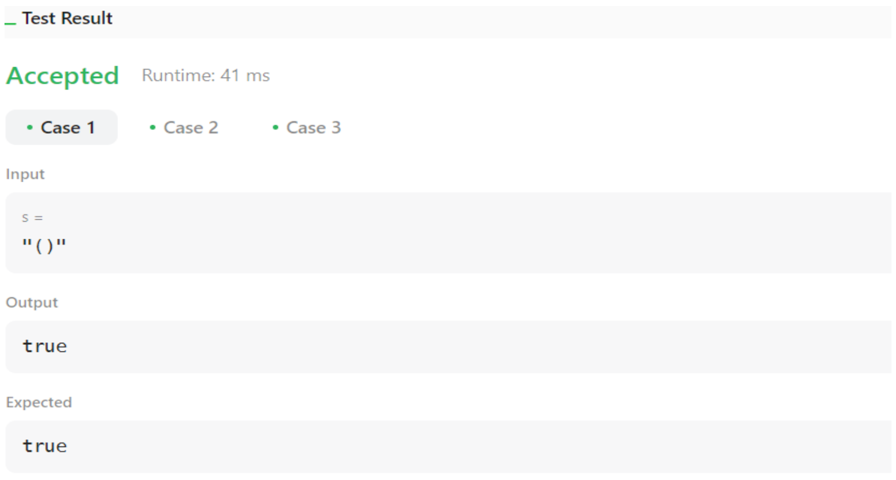
1.VALID PARENTHESES [20]

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.
An input string is valid if:
1.Open brackets must be closed by the same type of brackets.
2.Open brackets must be closed in the correct order.
3.Every close bracket has a corresponding open bracket of the same type.

CODE:

```
class Solution:
    def isValid(self, s: str) -> bool:
        stack=[]
        for i in s:
            if i=='(':
                stack.append(')')
            elif i=='{':
                stack.append('}')
            elif i=='[':
                stack.append(']')
            elif not stack or stack.pop()!=i:
                return False
        return not stack
```

OUTPUT:



```
Accepted   Runtime: 41 ms

Input
s =
"()"

Output
true

Expected
true
```

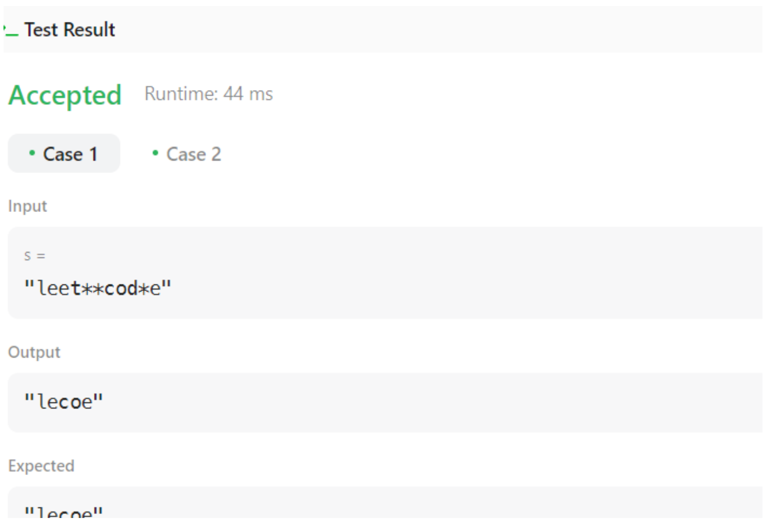
2. REMOVING STARS FROM A STRING [2390]

You are given a string s, which contains stars *.
In one operation, you can:
•Choose a star in s.
•Remove the closest non-star character to its left, as well as remove the star itself.
Return the string after all stars have been removed.
Note:
•The input will be generated such that the operation is always possible.
•It can be shown that the resulting string will always be unique.

CODE:

```
class Solution:
    def removeStars(self, s: str) -> str:
        ans=[]
        for i in s:
            if i=='*':
                ans.pop()
            else:
                ans+=[i]
        return "".join(ans)
```

OUTPUT:


Test Result
Accepted Runtime: 44 ms
Case 1 Case 2
Input
s =
```
"leet**cod*e"
```
Output
```
"lecoe"
```
Expected
```
"lecoe"
```

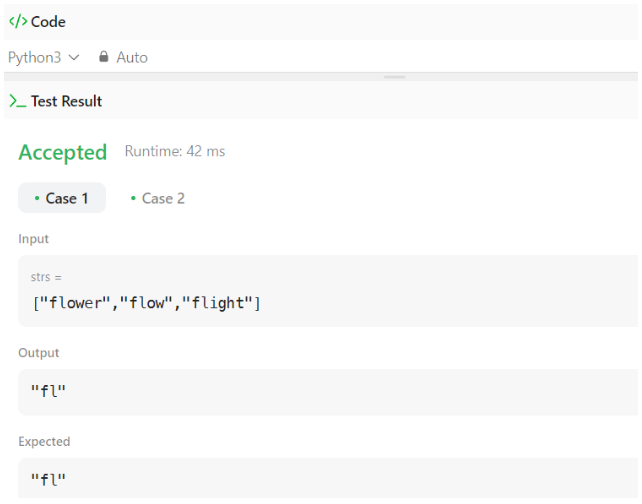
3. LONGEST COMMON PREFIX [14]

Write a function to find the longest common prefix string amongst an array of strings.
If there is no common prefix, return an empty string "".

CODE:

```
class Solution:
    def longestCommonPrefix(self, strs: List[str]) -> str:
        ans=""
        strs=sorted(strs)
        first=strs[0]
        last=strs[-1]
        for i in range(min(len(first), len(last))):
            if (first[i]!=last[i]):
                return ans
            ans+=first[i]
        return ans
```

OUTPUT:



Input

```
strs =
["flower","flow","flight"]
```

Output

```
"fl"
```

Expected

```
"fl"
```

4. MAXIMUM NESTING DEPTH OF THE PARENTHESES [1614]

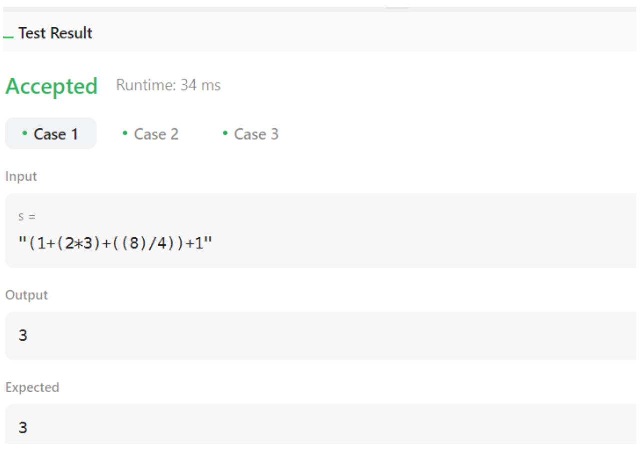
Given a valid parentheses string s, return the nesting depth of s. The nesting depth is the maximum number of nested parentheses.

CODE:

```
class Solution:
    def maxDepth(self, s: str) -> int:
```

```
    max_depth=0
    current_depth=0
    for char in s:
        if char=='(':
            current_depth+=1
            max_depth=max(max_depth,current_depth)
        elif char == ')':
            current_depth -=1
    return max_depth
```

OUTPUT:



Test Result

Accepted Runtime: 34 ms

Case 1 Case 2 Case 3

Input

```
s =
"(1+(2*3)+((8)/4))+1"
```

Output

```
3
```

Expected

```
3
```

Dt:08-06-2024

5. NUMBER OF STUDENTS UNABLE TO EAT LUNCH [1700]

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.
The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack. At each step:
If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue.
Otherwise, they will leave it and go to the queue's end.
This continues until none of the queue students want to take the top sandwich and are thus unable to eat.
You are given two integer arrays students and sandwiches where sandwiches[i] is the type of the ith sandwich in the stack (i = 0 is the top of the stack) and students[j] is the preference of the jth student in the initial

queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

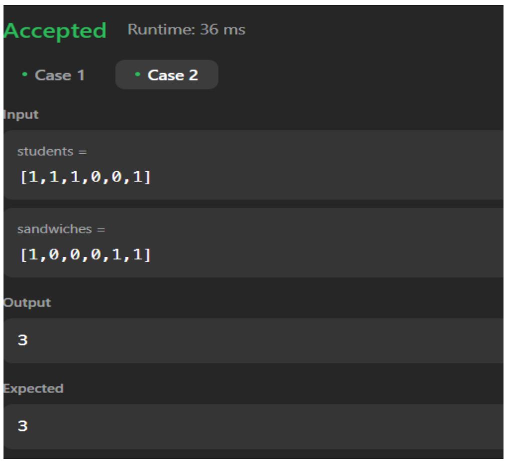
CODE:

```
class Solution:
    def countStudents(self, student: List[int], sandwitch: List[int]) -> int:
        count = Counter(student)
        for s in sandwitch:
            if count[s] > 0:
                count[s] -= 1
            else:
                return count.total()
        return 0
```

OR

```
class Solution:

    def countStudents(self, students: List[int], sandwiches: List[int]) -> int:

        count = len(students)

        while(sandwiches and students and sandwiches[0] in students):

            if(sandwiches[0]!=student[0]):

                students.append(students[0])

                students.pop(0)

            else:

                students.pop(0)

                sandwiches.pop(0)

                count-=1

        return count
```

OUTPUT:



```
Accepted    Runtime: 36 ms
• Case 1   • Case 2
Input
students =
[1,1,1,0,0,1]
sandwiches =
[1,0,0,0,1,1]
Output
3
Expected
3
```

6. REMOVE OUTERMOST PARENTHESES [1021]

A valid parentheses string is either empty "", "(" + A + ")", or A + B, where A and B are valid parentheses strings, and + represents string concatenation.

For example, "", "()", "(())()", and "(()(()))" are all valid parentheses strings. A valid parentheses string s is primitive if it is nonempty, and there does not exist a way to split it into s = A + B, with A and B nonempty valid parentheses strings.
Given a valid parentheses string s, consider its primitive decomposition: s = P1 + P2 + ... + Pk, where Pi are primitive valid parentheses strings.
Return s after removing the outermost parentheses of every primitive string in the primitive decomposition of s.

CODE:

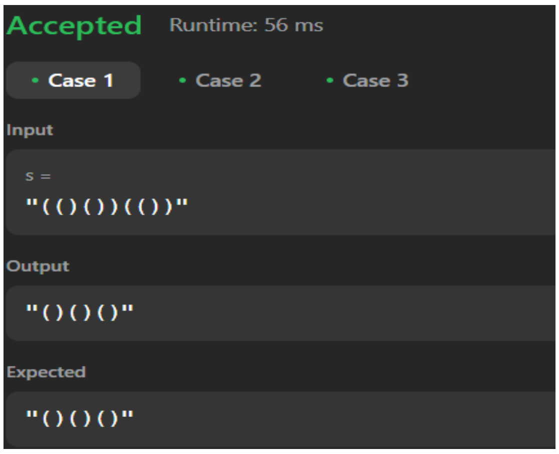
```
class Solution:
    def removeOuterParentheses(self, s: str) -> str:
        string = ""
        opened = 0
        for i in s:
            if i == "(":
                opened += 1
                if opened > 1:
                    string += i
            else:
                opened -= 1
                if opened > 0:
```

```
                string += i
        return string
```

OR

```
class Solution:
    def removeOuterParentheses(self, s: str) -> str:
        stack = []
        result = ""
        for char in s:
          if char=='(':
            if stack:
              result+=char
            stack.append(char)
          else:
            stack.pop()
            if stack:
              result+=char
        return result
```

OUTPUT:



Accepted Runtime: 56 ms

Case 1 Case 2 Case 3

Input

```
s =
"(()())(())"
```

Output

```
"()()()"
```

Expected

```
"()()()"
```

Dt:10-06-2024

7. NUMBER OF GOOD PAIRS [1512]

Given an array of integers nums, return the number of good pairs.
A pair (i, j) is called good if nums[i] == nums[j] and i < j.

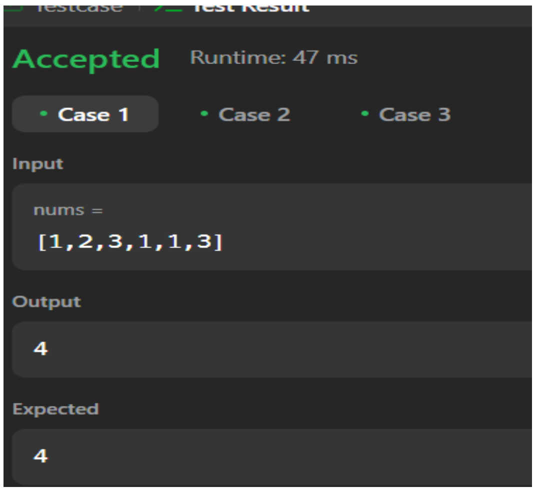
CODE:

```
class Solution:
    def numIdenticalPairs(self, nums: List[int]) -> int:

        ans = 0

        for i in range(len(nums)-1):# i is indexing from 0 to n-1

            for j in range(i+1,len(nums)):# j is from i+1 to n

                if nums[i] == nums[j]:# i < j always

                    ans +=1

        return ans
```

OR

```
class Solution:
    def numIdenticalPairs(self, nums: List[int]) -> int:
      hashMap={}
      res = 0
      for number in nums:
        if number in hashMap:
          res+=hashMap[number]
          hashMap[number]+=1
        else:
          hashMap[number]=1
      return res
```

OUTPUT:



Accepted Runtime: 47 ms

Case 1 Case 2 Case 3

```
Input
nums =
[1,2,3,1,1,3]
Output
4
Expected
4
```

8. KIDS WITH GREATEST NUMBER OF CANDIES [1431]

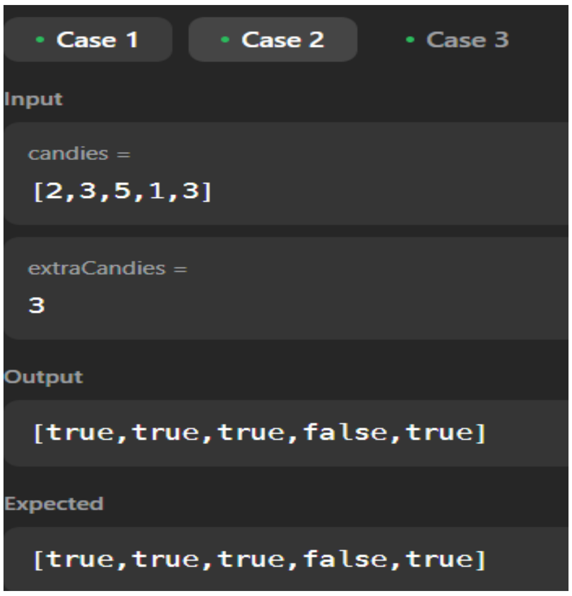
There are n kids with candies. You are given an integer array candies, where each candies[i] represents the number of candies the ith kid has, and an integer extraCandies, denoting the number of extra candies that you have. Return a boolean array result of length n, where result[i] is true if, after giving the ith kid all the extraCandies, they will have the greatest number of candies among all the kids, or false otherwise.
Note that multiple kids can have the greatest number of candies.
CODE:

```
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies: int) -> List[bool]:
        max_candies = max(candies)  # Find the maximum candies any kid has
        result = []  # Initialize the result list
                for candy in candies:
            # Check if giving the current kid all the extra candies makes them have 
the most candies
            if candy + extraCandies >= max_candies:
                result.append(True)  # If true, append True to result
            else:
                result.append(False)  # Otherwise, append False to result
        return result  # Return the result list
```

OR

```
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies: int) -> List[bool]:
      result_lst=[]
      max_num=max(candies)
      for num in candies:
        result_lst.append(num+extraCandies>=max_num)
      return result_lst
```

OUTPUT:



Input

candies =

```
[2,3,5,1,3]
```

extraCandies =

```
3
```

Output

```
[true,true,true,false,true]
```

Expected

```
[true,true,true,false,true]
```

Dt:11-06-2024

9. TWO SUM [01]

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.

CODE:

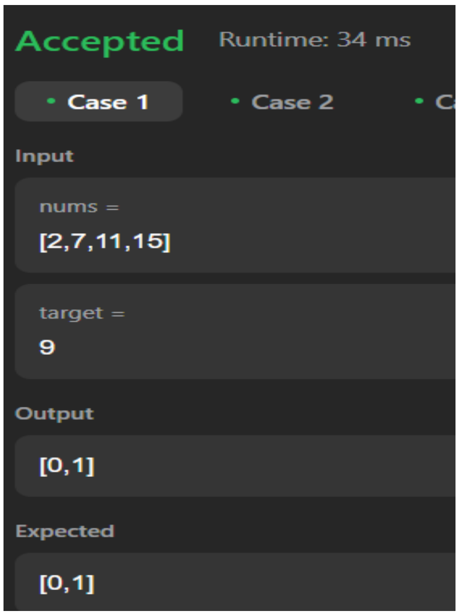
```
class Solution:
  def twoSum(self, nums: List[int], target: int) -> List[int]:
    n=len(nums)
    for i in range(n-1):
      for j in range(i+1,n):
        if nums[i]+nums[j]==target:
          return[i,j]
    return []
```

OR

```
class Solution:
  def twoSum(self, nums: List[int], target: int) -> List[int]:
    seen={}
```

```
   for i in range(len(nums)):
     diff=target-nums[i]
     if diff in seen:
       return[seen[diff],i]
     else:
       seen[nums[i]]=i
```

OUTPUT:



```
Accepted    Runtime: 34 ms

Case 1    Case 2

Input
nums =
[2,7,11,15]
target =
9

Output
[0,1]

Expected
[0,1]
```

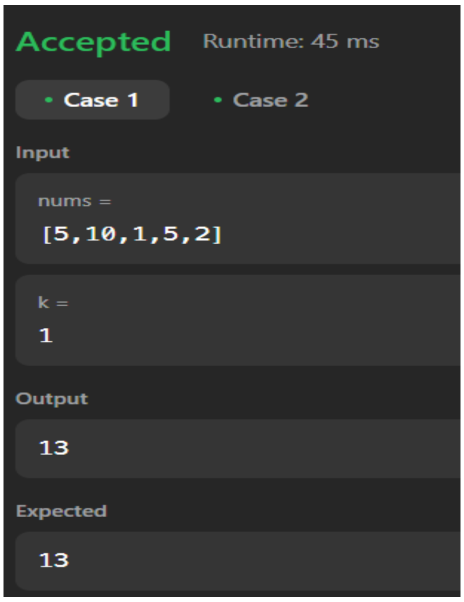
10. SUM OF VALUES AT INDICES WITH k SET BITS [2859]

You are given a 0-indexed integer array nums and an integer k.
Return an integer that denotes the sum of elements in nums whose corresponding indices have exactly k set bits in their binary representation.
The set bits in an integer are the 1's present when it is written in binary.

CODE:

```
class Solution:
    def sumIndicesWithKSetBits(self, nums: List[int], k: int) -> int:
        sum=0
        for i in range(len(nums)):
          if(bin(i).count('1')==k):
            sum=sum+nums[i]
        return sum
```

OUTPUT:



```
Accepted    Runtime: 45 ms
• Case 1    • Case 2
Input
nums =
[5,10,1,5,2]
k =
1
Output
13
Expected
13
```

11. HOW MANY NUMBERS ARE SMALLER THAN THE CURRENT NUMBER [1365]

Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i and nums[j] < nums[i].
Return the answer in an array.

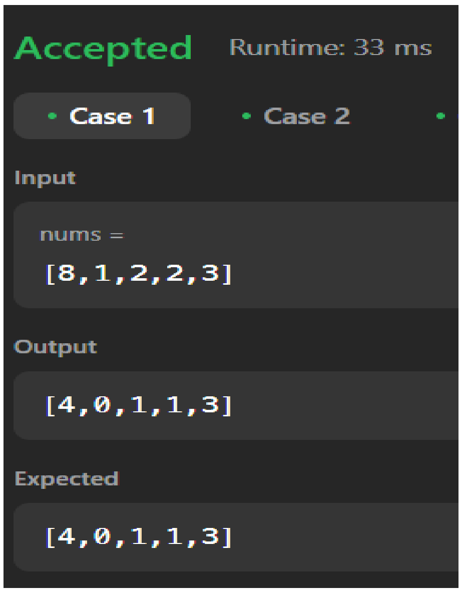
CODE:

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: List[int]) -> List[int]:
        n = []
        for i in range(len(nums)):
            count = 0
            for j in range(len(nums)):
                if i != j and nums[i] > nums[j]:
                    count += 1
            n.append(count)
        return n
```

OR

```
from typing import List

class Solution:
 def smallerNumbersThanCurrent(self, nums: List[int]) -> List[int]:
     result = []
     seen = {}
     temp = sorted(nums)
     for i in range(len(temp)):
         if temp[i] not in seen:
             seen[temp[i]] = i
     for num in nums:
         result.append(seen[num])
     return result
```

OUTPUT:



Accepted Runtime: 33 ms

Case 1 Case 2

Input

```
nums =
[8,1,2,2,3]
```

Output

```
[4,0,1,1,3]
```

Expected

```
[4,0,1,1,3]
```

12. LEFT AND RIGHT SUM DIFFERENCES [2574]

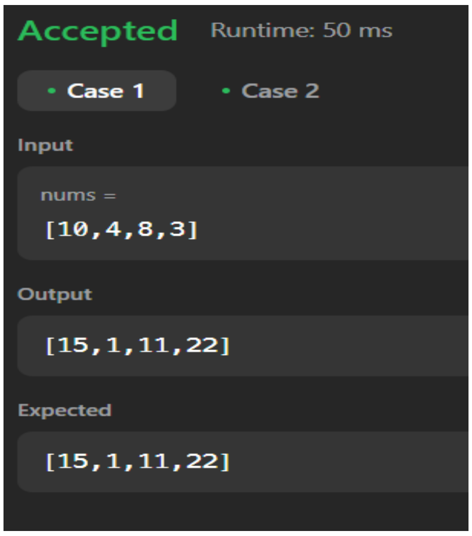
Given a 0-indexed integer array nums, find a 0-indexed integer array answer where:
answer.length == nums.length.
answer[i] = |leftSum[i] - rightSum[i]|.
Where:
leftSum[i] is the sum of elements to the left of the index i in the array nums. If there is no such element, leftSum[i] = 0.
rightSum[i] is the sum of elements to the right of the index i in the array

nums. If there is no such element, rightSum[i] = 0.
Return the array answer.

CODE:

```
class Solution:
    def leftRightDifference(self, nums: List[int]) -> List[int]:
      ans=[]
      for i in range(len(nums)):
        ans.append(abs(sum(nums[:i])-sum(nums[i+1:])))
      return ans
```

OUTPUT:



```
Accepted    Runtime: 50 ms
• Case 1    • Case 2
Input
nums =
[10,4,8,3]
Output
[15,1,11,22]
Expected
[15,1,11,22]
```

13. SHUFFLE THE ARRAY [1470]

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].
Return the array in the form [x1,y1,x2,y2,...,xn,yn].

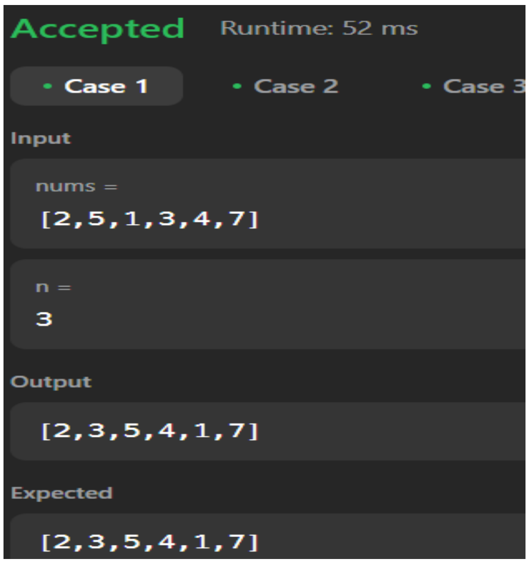
CODE:

```
class Solution:
    def shuffle(self, nums: List[int], n: int) -> List[int]:
      l=[]
      for i in range(n):
        l+=[nums[i]]
        l+=[nums[i+n]]
      return l
```

OR

```
class Solution:
    def shuffle(self, nums: List[int], n: int) -> List[int]:
        l = []
        for i in range(0,n):
            l.append(nums[i])
            l.append(nums[i+n])
        return l
```

OUTPUT:



Input

nums =

```
[2,5,1,3,4,7]
```

n =

```
3
```

Output

```
[2,3,5,4,1,7]
```

Expected

```
[2,3,5,4,1,7]
```

14. UNIQUE MORSE CODE WORDS [804]

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:
'a' maps to ".-",
'b' maps to "-...",
'c' maps to "-.-.", and so on.
For convenience, the full table for the 26 letters of the English alphabet is given below:
[".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."]
Given an array of strings words where each word can be written as a concatenation of the Morse code of each letter.

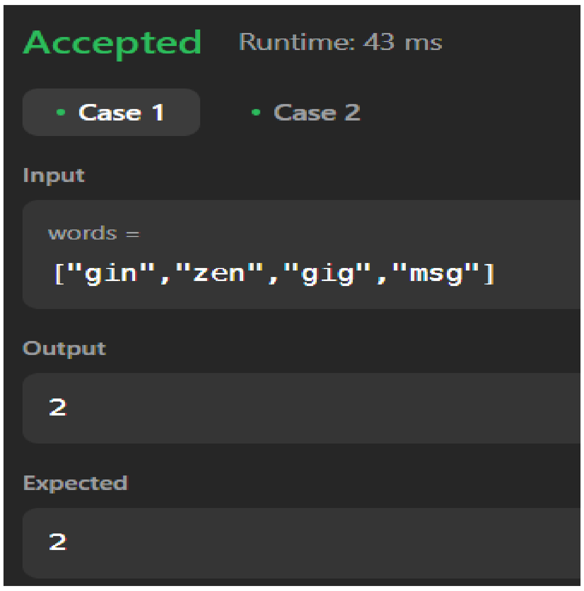
CODE:

```
class Solution:
    def uniqueMorseRepresentations(self, words: List[str]) -> int:
        morse = {'a':".-", 'b':"-...", 'c':"-.-.", 'd':"-..", 'e':".", 'f':"..-.", 'g':"--.", 'h':"....",
'i':"..", 'j':".---", 'k':"-.-", 'l':".-..", 'm':"--", 'n':"-.", 'o':"---", 'p':".--.", 'q':"--.-", 'r':".-
.", 's':"...", 't':"-", 'u':"..-", 'v':"...-", 'w':".--", 'x':"-..-", 'y':"-.--", 'z':"--.."}
        s = set()
        for word in words:
            temp = ""
            for i in range(len(word)):
                temp += morse[word[i]]
            s.add(temp)
        return len(s)
```

OR

```
class Solution:

    def uniqueMorseRepresentations(self, words: List[str]) -> int:

       letters="abcdefghijklmnopqrstuvwxyz"

       moorse_code=["".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--
","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--..""]

       morse_dict=dict(zip(letters,morse_code))

       words2=[]

       for word in words:

        k=""

        for i in word:

            k+=morse_dict[i]

        words2.append(k)

    return len(set(words2))
```

OUTPUT:



Accepted Runtime: 43 ms

Input

```
words =
["gin","zen","gig","msg"]
```

Output

```
2
```

Expected

```
2
```

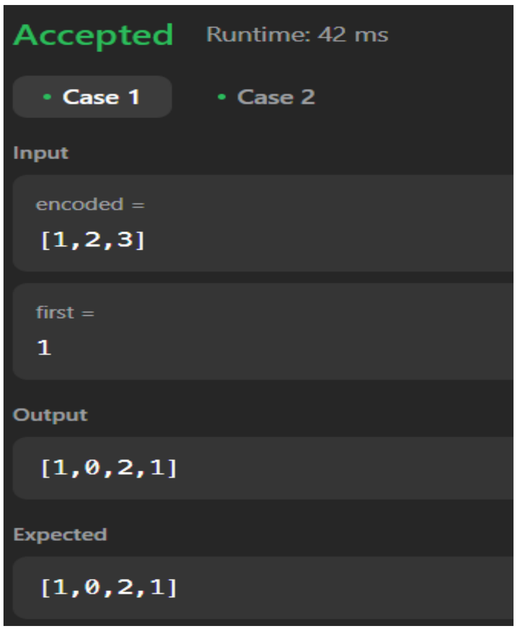
15. DECODE XORED ARRAY [1720]

There is a hidden integer array arr that consists of n non-negative integers. It was encoded into another integer array encoded of length n - 1, such that encoded[i] = arr[i] XOR arr[i + 1]. For example, if arr = [1,0,2,1], then encoded = [1,2,3].
You are given the encoded array. You are also given an integer first, that is the first element of arr, i.e. arr[0].
Return the original array arr. It can be proved that the answer exists and is unique.

CODE:

```
class Solution:
   def decode(self, A: List[int], first: int) -> List[int]:
      ans = [first]
      n = len(A)
      for i in range(n):
         a = ans[-1] ^ A[i]
         ans.append(a)
      return ans
```

OUTPUT:



```
Input
encoded =
[1,2,3]
first =
1
Output
[1,0,2,1]
Expected
[1,0,2,1]
```

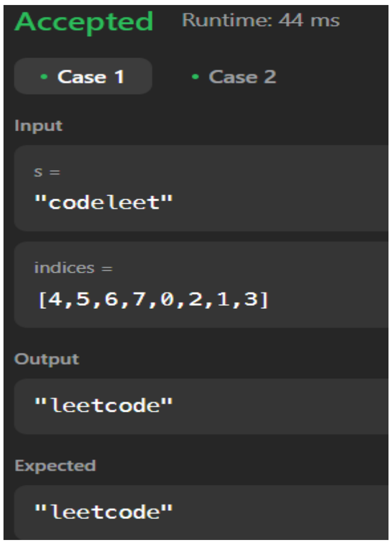
16. SHUFFLE STRING [1528]

You are given a string s and an integer array indices of the same length. The string s will be shuffled such that the character at the ith position moves to indices[i] in the shuffled string.
Return the shuffled string.

CODE:

```
class Solution:
    def restoreString(self, s: str, indices: List[int]) -> str:
        C=""
        indices=list(indices)
        for i in range(len(s)):
            b=indices.index(i)
            C+=s[b]
        return C
```

OUTPUT:



```
Accepted    Runtime: 44 ms
Case 1    Case 2
Input
s =
"codeleet"
indices =
[4,5,6,7,0,2,1,3]
Output
"leetcode"
Expected
"leetcode"
```

Dt:12-06-2024

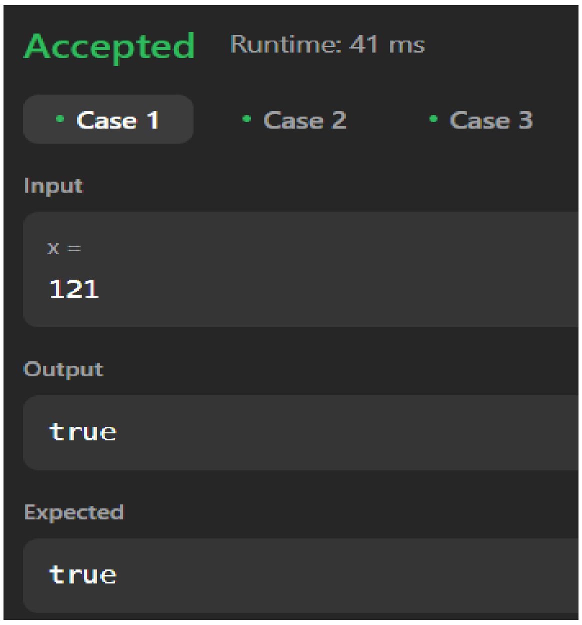
17. PALINDROME NUMBERS [09]

Given an integer x, return true if x is a palindrome, and false otherwise.

CODE:

```
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x < 0:
            return False
        n = 0
        temp = x
        while temp != 0:
            digit = temp % 10
            n = n * 10 + digit
            temp //= 10
        return n == x
```

OUTPUT:



```
Accepted   Runtime: 41 ms
• Case 1   • Case 2   • Case 3
Input
x =
121
Output
true
Expected
true
```

18. MERGE TWO SORTED LISTS [21]

You are given the heads of two sorted linked lists list1 and list2.
Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.
Return the head of the merged linked list.

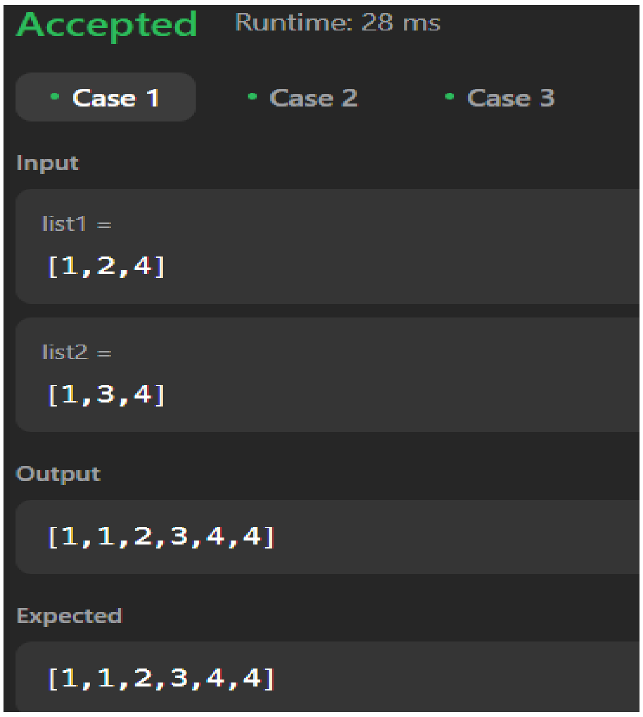
CODE:

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
   def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) ->
Optional[ListNode]:
      merge = ListNode()
      node = merge
      while list1 and list2:
         if list1.val <= list2.val:
            node.next = ListNode(list1.val)
            list1 = list1.next
         else:
            node.next = ListNode(list2.val)
            list2 = list2.next
        node = node.next
    node.next = list1 or list2
    return merge.next
```

OR

```
from typing import Optional

class ListNode:

  def __init__(self, val=0, next=None):

    self.val = val

    self.next = next

class Solution:

  def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) ->
Optional[ListNode]:

    newHead = current = ListNode()

    while list1 and list2:

      if list1.val < list2.val:

        current.next = list1

        list1 = list1.next

      else:

        current.next = list2

        list2 = list2.next

      current = current.next

    current.next = list1 if list1 else list2

    return newHead.next
```

OUTPUT:



Input

list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

Output

```
[1,1,2,3,4,4]
```

Expected

```
[1,1,2,3,4,4]
```

19. SORT COLOURS [75]

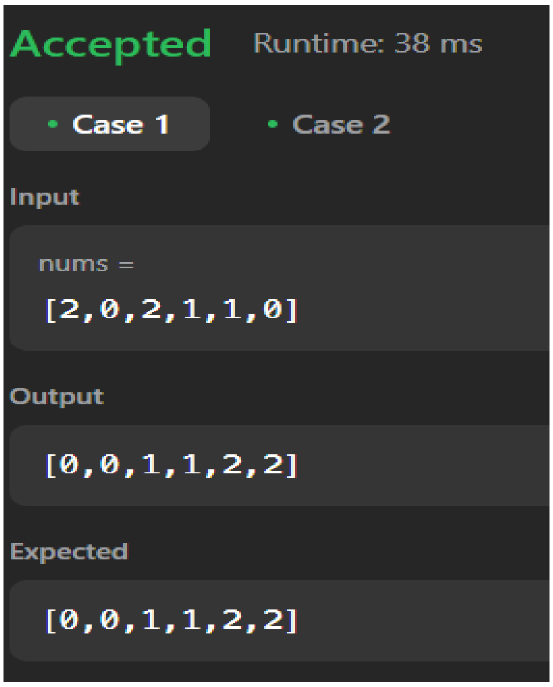
Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.
We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.
You must solve this problem without using the library's sort function.

CODE:

```
class Solution:
    def sortColors(self, nums: List[int]) -> None:
        r, w, b = 0, 0, len(nums) - 1
        while w <= b:
            if nums[w] == 0:
                nums[r], nums[w] = nums[w], nums[r]
                r += 1
                w += 1
            elif nums[w] == 1:
                w += 1
            else:
```

```
            nums[w], nums[b] = nums[b], nums[w]
            b -= 1
```

OUTPUT:



```
Accepted   Runtime: 38 ms

• Case 1   • Case 2

Input
nums =
[2,0,2,1,1,0]

Output
[0,0,1,1,2,2]

Expected
[0,0,1,1,2,2]
```

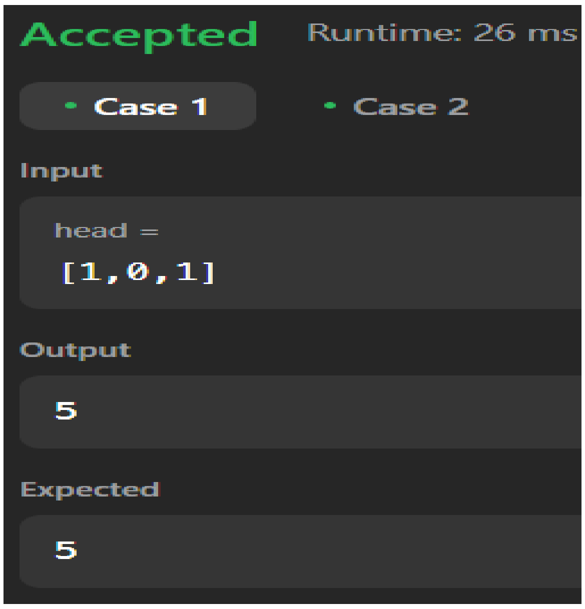
20. CONVERT BINARY NUMBER IN A LINKED LIST TO INTEGER [1290]

Given head which is a reference node to a singly-linked list. The value of each node in the linked list is either 0 or 1. The linked list holds the binary representation of a number.
Return the decimal value of the number in the linked list.
The most significant bit is at the head of the linked list.

CODE:

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def getDecimalValue(self, head: ListNode) -> int:
        s = ''
        if head is None:
            return
        temp = head
        while temp is not None:
            s = s + str(temp.val)
            temp = temp.next
        return int(s,2)
```

OUTPUT:



```
Accepted    Runtime: 26 ms
• Case 1    • Case 2
Input
head =
[1,0,1]
Output
5
Expected
5
```

21. MIDDLE OF THE LINKED LIST [876]

Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the second middle node.

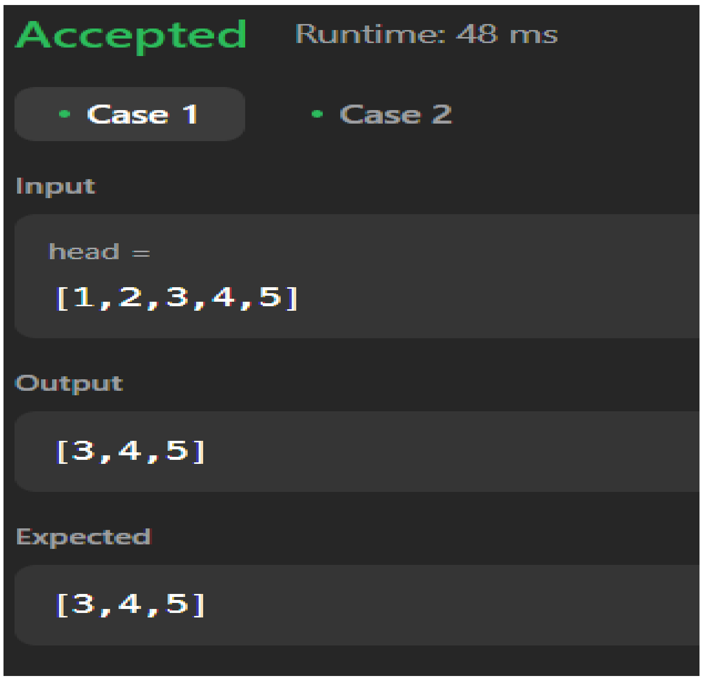
CODE:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val=val
        self.next=next
class solution:
    def middleNode(self, head: ListNode)->ListNode:
        count=0
        current=head
        while current:
            count+=1
            current=current.next

        current=head

        for i in range(count//2):

            current=current.next

        return current
```

OR

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def middleNode(self, head: Optional[ListNode]) -> Optional[ListNode]:
        slow_pointer = head
        fast_pointer = head
        while fast_pointer is not None and fast_pointer.next is not None:
            slow_pointer = slow_pointer.next
            fast_pointer = fast_pointer.next.next
        return slow_pointer
```

OUTPUT:



```
Accepted    Runtime: 48 ms

• Case 1    • Case 2

Input
head =
[1,2,3,4,5]

Output
[3,4,5]

Expected
[3,4,5]
```

22. ROMAN TO INTEGER [13]

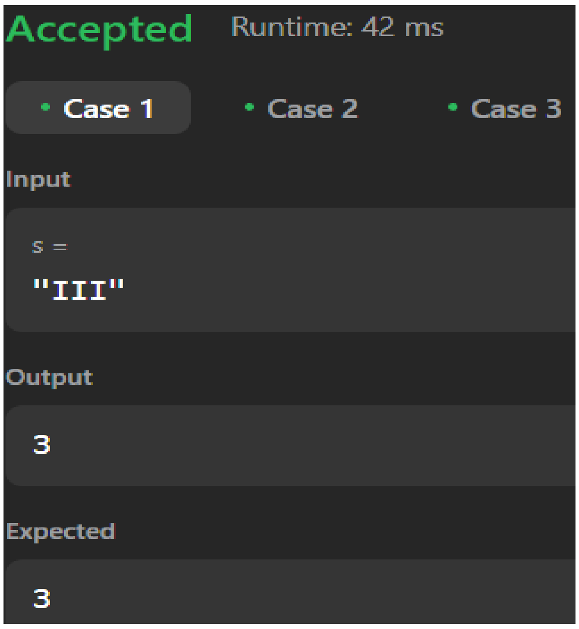
Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	**Value**
I	**1**
V	**5**
X	**10**
L	**50**
C	**100**
D	**500**
M	**1000**

CODE:

```
class Solution:
    def romanToInt(self, s: str) -> int:
        roman={"I":1,"V":5,"X":10,"L":50,"C":100,"D":500,"M":1000}
        number=0
        for i in range(len(s)-1):
            if roman[s[i]] < roman[s[(i+1)]]:
                number-=roman[s[i]]
            else:
                number+=roman[s[i]]
        return number+roman[s[-1]]
```

OUTPUT:



```
Accepted    Runtime: 42 ms

• Case 1    • Case 2    • Case 3

Input
s =
"III"

Output
3

Expected
3
```

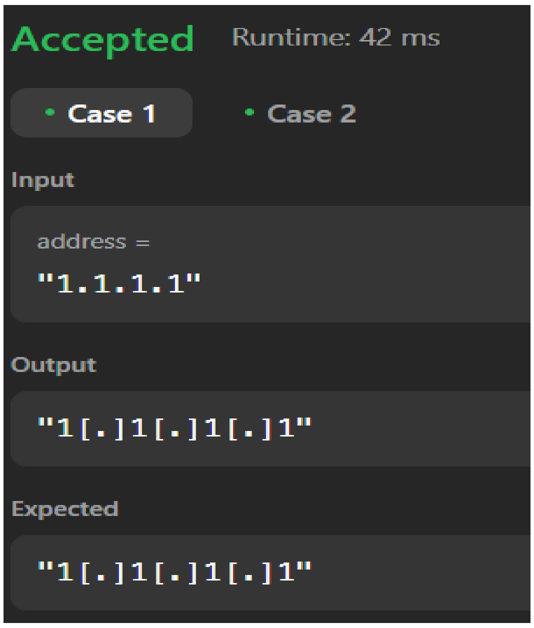
23. DEFANGING AN IP ADDRESS [1108]

Given a valid (IPv4) IP address, return a defanged version of that IP address. A defanged IP address replaces every period "." with "[.]".

CODE:

```
class Solution:
    def defangIPaddr(self, address: str) -> str:
        ans=address.replace(".","[.]")
        return ans
```

OUTPUT:



```
Accepted    Runtime: 42 ms
• Case 1    • Case 2
Input
address =
"1.1.1.1"
Output
"1[.]1[.]1[.]1"
Expected
"1[.]1[.]1[.]1"
```

24. MINIMUM COST TO REACH DESTINATION IN TIME [1928]

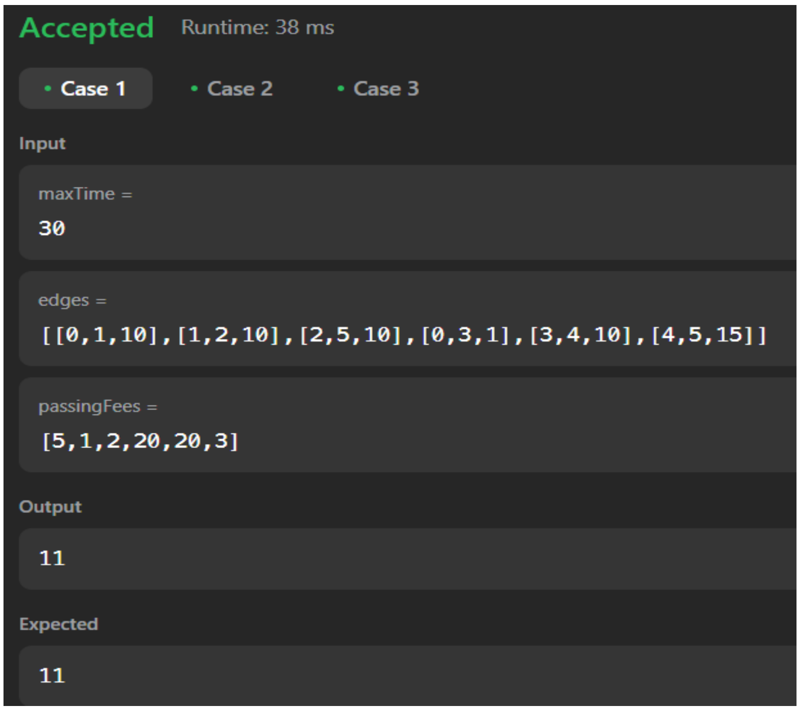
There is a country of n cities numbered from 0 to n - 1 where all the cities are connected by bi-directional roads. The roads are represented as a 2D integer array edges where edges[i] = [xi, yi, timei] denotes a road between cities xi and yi that takes timei minutes to travel. There may be multiple roads of differing travel times connecting the same two cities, but no road connects a city to itself.
Each time you pass through a city, you must pay a passing fee. This is represented as a 0-indexed integer array passingFees of length n where passingFees[j] is the amount of dollars you must pay when you pass through city j.

In the beginning, you are at city 0 and want to reach city n - 1 in maxTime minutes or less. The cost of your journey is the summation of passing fees for each city that you passed through at some moment of your journey (including the source and destination cities).
Given maxTime, edges, and passingFees, return the minimum cost to complete your journey, or -1 if you cannot complete it within maxTime minutes.

CODE:

```
class Solution:
   def minCost(self, maxTime: int, edges: List[List[int]], passingFees: List[int]) ->
int:
      n = len(passingFees)
      # Create adjacency list representation of the graph
      graph = [[] for _ in range(n)]
      for u, v, time in edges:
         graph[u].append((v, time))
         graph[v].append((u, time))
            # Dijkstra's Algorithm
      pq = [(passingFees[0], 0, 0)]  # (total fees, time, city)
      dist = {(0, 0): passingFees[0]}  # (time, city): total fees
            while pq:
         total_fees, time, city = heapq.heappop(pq)
                  if city == n - 1:
            return total_fees
         for neighbor, travel_time in graph[city]:
            new_time = time + travel_time
            if new_time <= maxTime:
               new_fees = total_fees + passingFees[neighbor]
               if (new_time, neighbor) not in dist or new_fees < dist[(new_time,
neighbor)]:
                  dist[(new_time, neighbor)] = new_fees
                  heapq.heappush(pq, (new_fees, new_time, neighbor))
            return -1  # If destination is unreachable within maxTime
```

OUTPUT:


```
Accepted   Runtime: 38 ms
• Case 1   • Case 2   • Case 3
Input
maxTime =
30
edges =
[[0,1,10],[1,2,10],[2,5,10],[0,3,1],[3,4,10],[4,5,15]]
passingFees =
[5,1,2,20,20,3]
Output
11
Expected
11
```

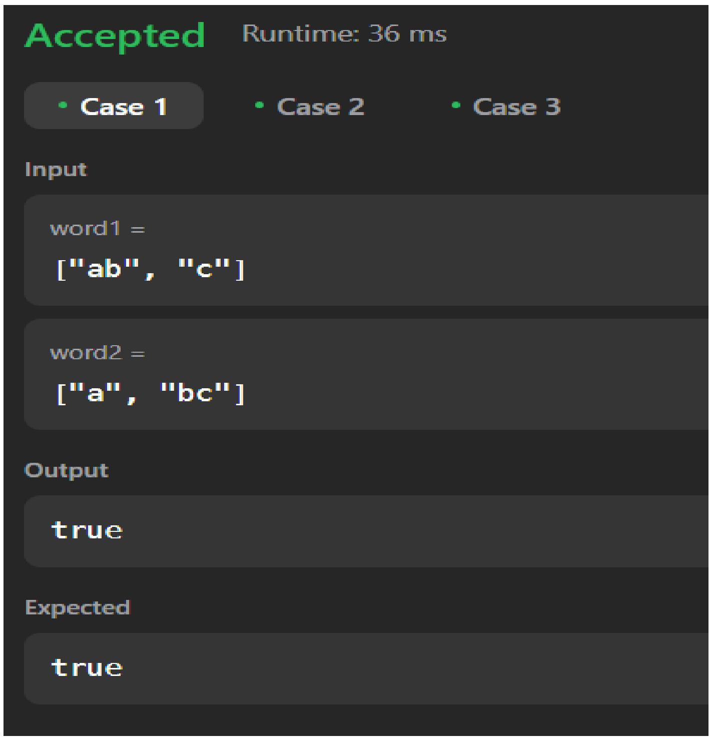
25. CHECK IF TWO STRINGS ARRAYS ARE EQUIVALENT [1662]

Given two string arrays word1 and word2, return true if the two arrays represent the same string, and false otherwise.
A string is represented by an array if the array elements concatenated in order forms the string.

CODE:

```
class Solution:
   def arrayStringsAreEqual(self, word1: List[str], word2: List[str]) -> bool:
      joined_word1 = "".join(word1)
      joined_word2 = "".join(word2)
      if (joined_word1==joined_word2):
         return True
      else:
         return False
```

OUTPUT:


Accepted Runtime: 36 ms

Case 1 Case 2 Case 3

Input

```
word1 =
["ab", "c"]
```

```
word2 =
["a", "bc"]
```

Output

```
true
```

Expected

```
true
```

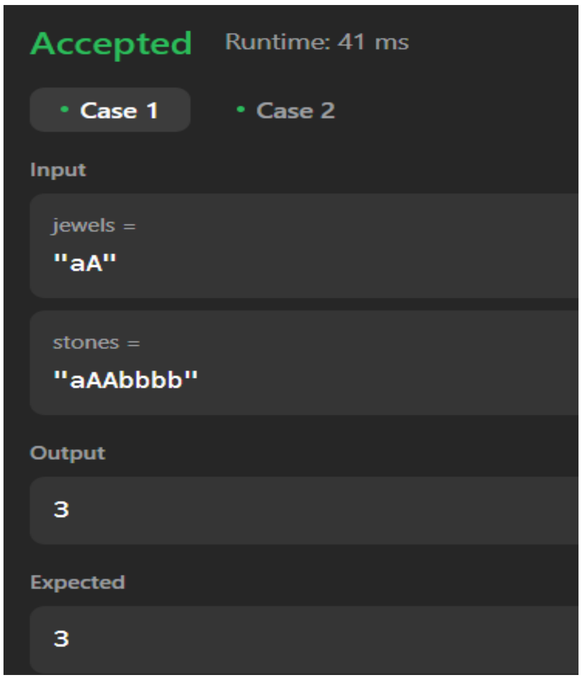
26. JEWELS AND STONES [771]

You're given strings jewels representing the types of stones that are jewels, and stones representing the stones you have. Each character in stones is a type of stone you have. You want to know how many of the stones you have are also jewels.
Letters are case sensitive, so "a" is considered a different type of stone from "A".

CODE:

```
class Solution:
    def numJewelsInStones(self, jewels: str, stones: str) -> int:
        jew_list=list(jewels)
        answer=0
        for j in jew_list:
         answer+=stones.count(j)
        return answer
```

OUTPUT:



```
Accepted   Runtime: 41 ms
• Case 1   • Case 2
Input
jewels =
"aA"
stones =
"aAAbbbb"
Output
3
Expected
3
```

27. FIND MAXIMUM NUMBER OF STRING PAIRS [2744]

You are given a 0-indexed array words consisting of distinct strings.
The string words[i] can be paired with the string words[j] if:
The string words[i] is equal to the reversed string of words[j].
0 <= i < j < words.length.
Return the maximum number of pairs that can be formed from the array words.
Note that each string can belong in at most one pair.

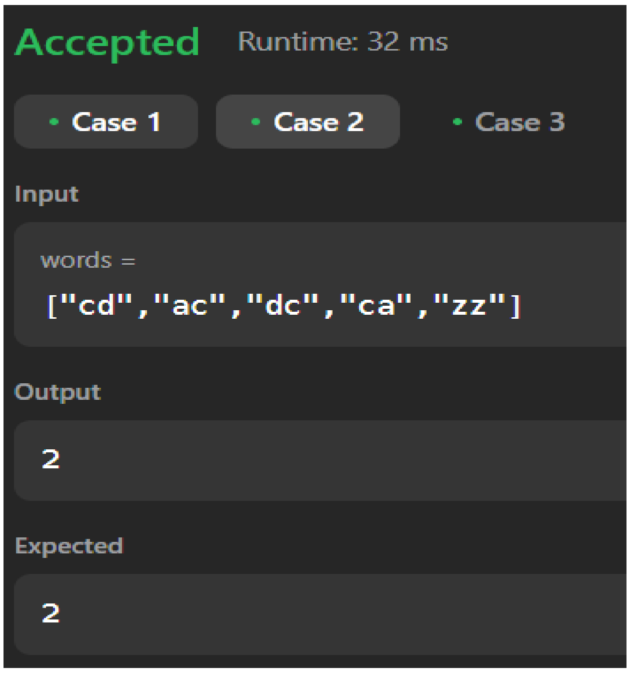
CODE:

```
class Solution:
    def maximumNumberOfStringPairs(self, words: List[str]) -> int:
        total = 0
        c = set()
        for i in range(len(words)):
            for j in range(i + 1, len(words)):
                word1 = words[i]
                word2 = words[j]
                if word1 == word2[::-1] and word1 not in c and word2 not in c:
                    total += 1
                    c.add(word1)
                c.add(word2)
        return total
```

OR

```
class Solution:
    def maximumNumberOfStringPairs(self, words: List[str]) -> int:
        total = 0
        for i in range(len(words)):
          if words[i][::-1] in (words[i+1:]):
            total+=1
        return total
```

OUTPUT:



```
Accepted    Runtime: 32 ms
• Case 1   • Case 2   • Case 3
Input
words =
["cd","ac","dc","ca","zz"]
Output
2
Expected
2
```

28. SORT THE PEOPLE [2418]

You are given an array of strings names, and an array heights that consists of distinct positive integers. Both arrays are of length n.

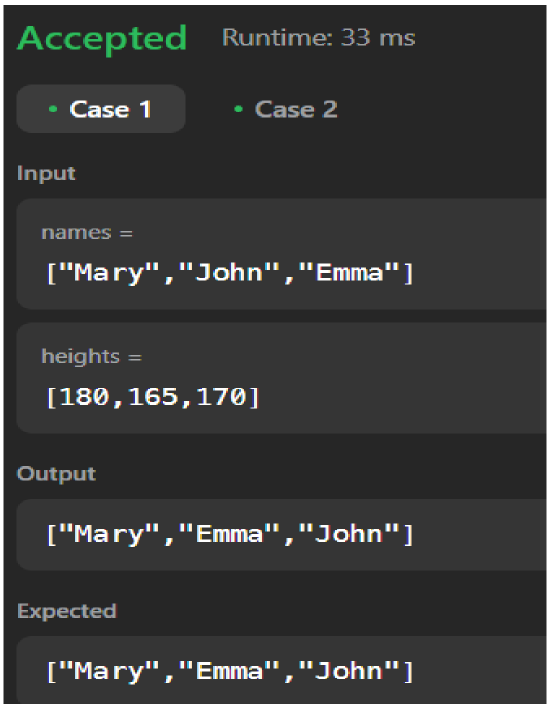
For each index i, names[i] and heights[i] denote the name and height of the ith person.

Return names sorted in descending order by the people's heights.

CODE:

```
class Solution:
    def sortPeople(self, names: List[str], heights: List[int]) -> List[str]:
        ans=zip(heights,names)
        l=[]
        for i, j in sorted(ans):
            l.append(j)
        return l[::-1]
```

OUTPUT:



```
Accepted    Runtime: 33 ms
• Case 1    • Case 2
Input
names =
["Mary","John","Emma"]
heights =
[180,165,170]
Output
["Mary","Emma","John"]
Expected
["Mary","Emma","John"]
```

29. SMALLEST EVEN MULTIPLE [2413]

Given a positive integer n, return the smallest positive integer that is a multiple of both 2 and n.

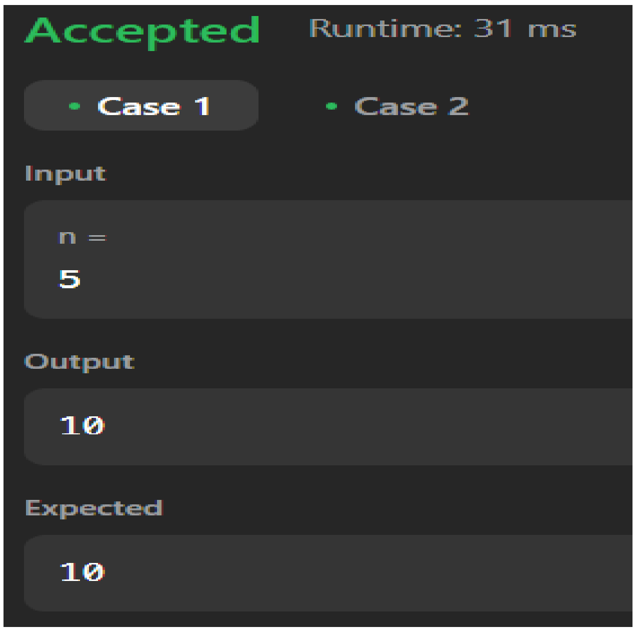
CODE:

```
class Solution:
    def smallestEvenMultiple(self, n):
        if n%2==0:
            return n
        else:
            i = n+n
            return i
```

OR

```
class Solution:
    def smallestEvenMultiple(self, n: int) -> int:
        if n%2==0:
            return n
        else:
            ans=n*2
            return ans
```

OUTPUT:



```
Accepted    Runtime: 31 ms
• Case 1    • Case 2
Input
n =
5
Output
10
Expected
10
```

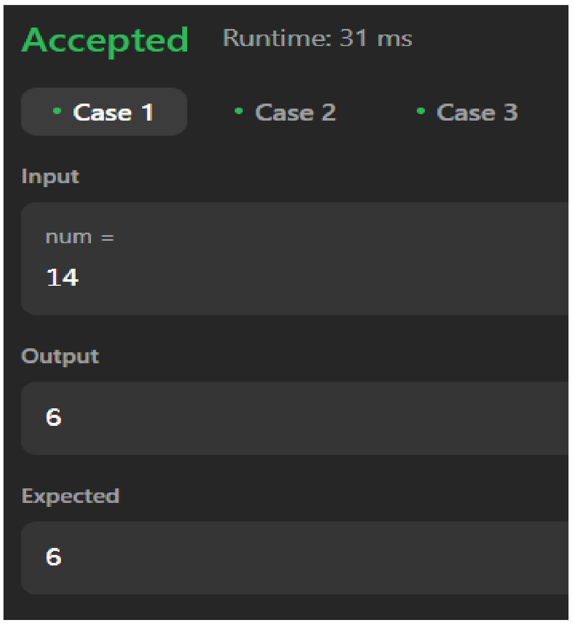
30. NUMBER OF STEPS TO REDUCE A NUMBER TO ZERO [1342]

Given an integer num, return the number of steps to reduce it to zero. In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

CODE:

```
class Solution:
    def numberOfSteps(self, n: int) -> int:
        step=0
        while n!=0:
            if n%2==0:
                n/=2
                step+=1
            else:
                n-=1
                step+=1
        return step
```

OUTPUT:



```
Accepted    Runtime: 31 ms
• Case 1    • Case 2    • Case 3
Input
num =
14
Output
6
Expected
6
```

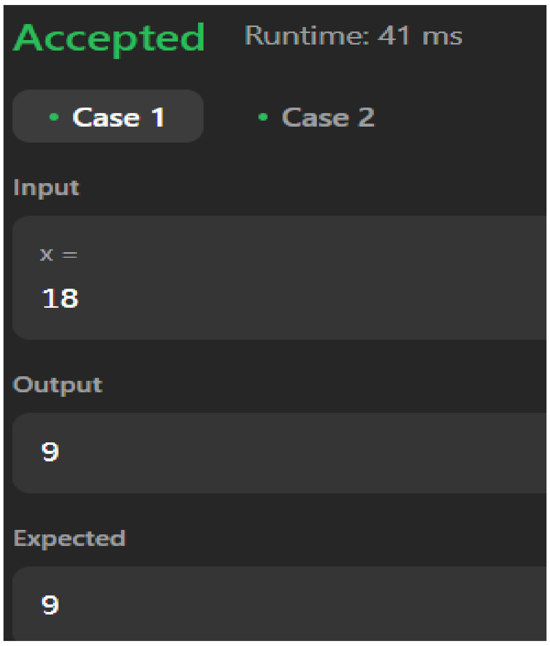
31. HARSHAD NUMBER [3099]

An integer divisible by the sum of its digits is said to be a Harshad number. You are given an integer x. Return the sum of the digits of x if x is a Harshad number, otherwise, return -1.

CODE:

```
class Solution:
    def sumOfTheDigitsOfHarshadNumber(self, x: int) -> int:
        result = 0
        temp = x
        while temp != 0:
            result += temp % 10
            temp=temp // 10
        if x % result == 0:
            return result
        else:
            return -1
```

OUTPUT:



```
Accepted    Runtime: 41 ms
• Case 1    • Case 2
Input
x =
18
Output
9
Expected
9
```

Dt:13-06-2024

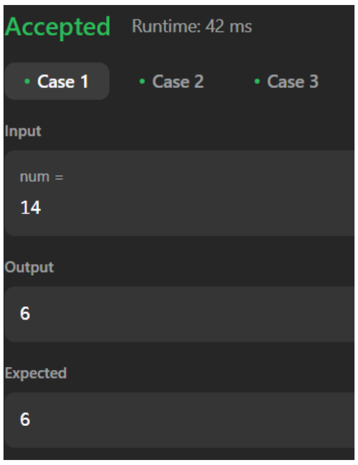
32. NUMBER OF STEPS TO REDUCE A NUMBER TO ZERO [1342]

Given an integer num, return the number of steps to reduce it to zero.
In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

CODE:

```
class Solution:
    def numberOfSteps(self, n: int) -> int:
        s=0
        while n!=0:
            if n%2==0:
                n/=2
                s+=1
            else:
                n-=1
                s+=1
        return s
```

OUTPUT:



```
Accepted   Runtime: 42 ms
• Case 1   • Case 2   • Case 3
Input
num =
14
Output
6
Expected
6
```

33. MINIMUM NUMBER OF MOVES TO SEAT EVERYONE [2037]

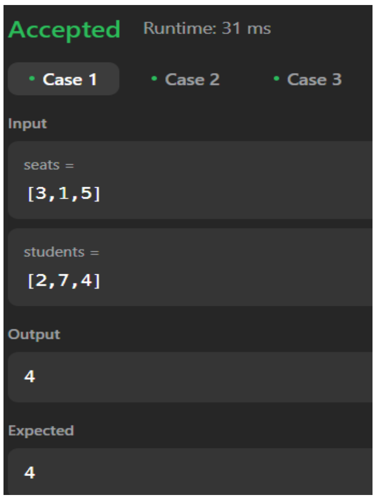
There are n seats and n students in a room. You are given an array seats of length n, where seats[i] is the position of the ith seat. You are also given the array students of length n, where students[j] is the position of the jth student.
You may perform the following move any number of times:

Increase or decrease the position of the ith student by 1 (i.e., moving the ith student from position x to x + 1 or x - 1)
Return the minimum number of moves required to move each student to a seat such that no two students are in the same seat.
Note that there may be multiple seats or students in the same position at the beginning.

CODE:

```
class Solution:
    def minMovesToSeat(self, seats: List[int], students: List[int]) -> int:
        seats.sort()
        students.sort()
        count = 0
        for i in range(len(seats)):
            count += abs(seats[i] - students[i])
        return count
```

OUTPUT:



```
Accepted    Runtime: 31 ms
Case 1   Case 2   Case 3
Input
seats =
[3,1,5]
students =
[2,7,4]
Output
4
Expected
4
```

Dt:14-06-2024

34. A NUMBER AFTER A DOUBLE REVERSAL [2119]

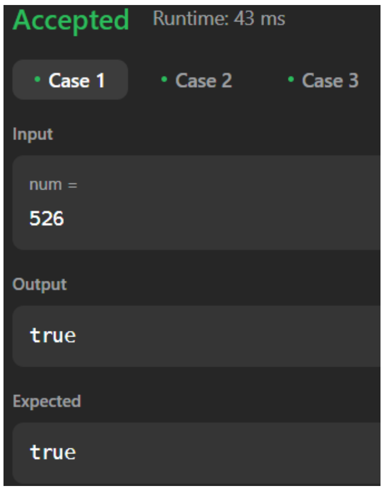
Reversing an integer means to reverse all its digits.
For example, reversing 2021 gives 1202. Reversing 12300 gives 321 as the leading zeros are not retained.
Given an integer num, reverse num to get reversed1, then reverse reversed1

to get reversed2. Return true if reversed2 equals num. Otherwise return false.

CODE:

```
class Solution:
    def isSameAfterReversals(self, num: int) -> bool:
        if num == 0 :
            return True
        if str(num)[-1] == "0" :
            return False
        return True
```

OUTPUT:



Accepted Runtime: 43 ms

• Case 1 • Case 2 • Case 3

Input

```
num =
526
```

Output

```
true
```

Expected

```
true
```

35. LARGEST ODD NUMBER IN STRING [1903]

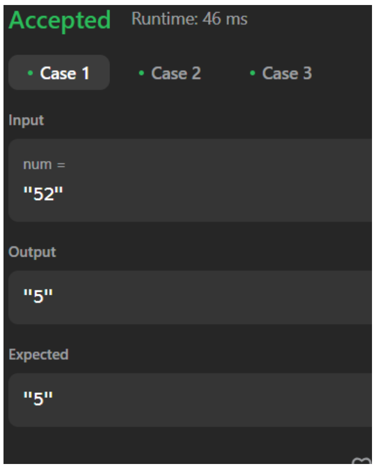
You are given a string num, representing a large integer. Return the largest-valued odd integer (as a string) that is a non-empty substring of num, or an empty string "" if no odd integer exists.
A substring is a contiguous sequence of characters within a string.

CODE:

```
class Solution:
    def largestOddNumber(self, num: str) -> str:
        h={"1","3","5","7","9"}
        ind=-1
        for i in range(len(num)-1,-1,-1):
            if num[i] not in h :continue
            ind=i
```

```
                break
        return num[:ind+1] if ind!=-1 else ""
```

OUTPUT:



```
Accepted   Runtime: 46 ms

• Case 1   • Case 2   • Case 3

Input
num =
"52"

Output
"5"

Expected
"5"
```