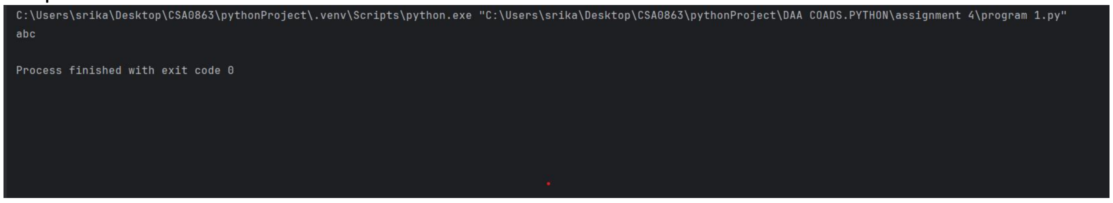
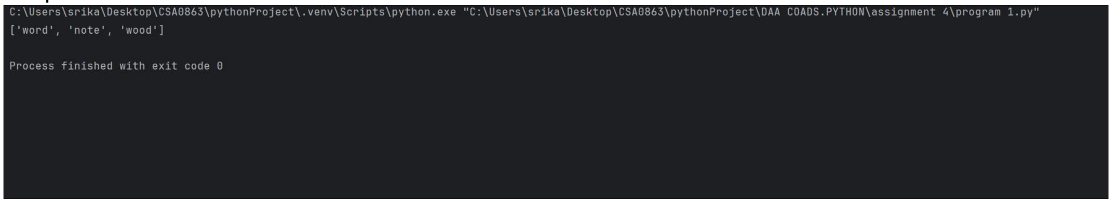
Odd String Difference
You are given an array of equal-length strings words. Assume that the length of each string is n.
Each string words[i] can be converted into a difference integer array difference[i] of length n - 1 where difference[i][j] = words[i][j+1] - words[i][j] where 0 <= j <= n - 2.
Note that the difference between two letters is the difference between their positions in the alphabet i.e. the position of 'a' is 0, 'b' is 1, and 'z' is 25.
For example, for the string "acb", the difference integer array is [2 - 0, 1 - 2] = [2, -1].
All the strings in words have the same difference integer array, except one. You should find that string.
Return the string in words that has different difference integer array.
Example 1:
Input: words = ["adc","wzy","abc"]
Output: "abc"
Explanation:
- The difference integer array of "adc" is [3 - 0, 2 - 3] = [3, -1].
- The difference integer array of "wzy" is [25 - 22, 24 - 25]= [3, -1].
- The difference integer array of "abc" is [1 - 0, 2 - 1] = [1, 1].
The odd array out is [1, 1], so we return the corresponding string, "abc".

```
Program:def oddStringDifference(words):
    def calculate_difference(word):
        return [ord(word[i + 1]) - ord(word[i]) for i in range(len(word) - 1)]
    differences = [calculate_difference(word) for word in words]
    count_map = {}
    for difference in differences:
        count_map[tuple(difference)] = count_map.get(tuple(difference), 0) + 1
    odd_difference = None
    for difference, count in count_map.items():
        if count % 2 == 1:
            odd_difference = difference
            break
    for i in range(len(words)):
        if tuple(differences[i]) == odd_difference:
            return words[i]
words = ["adc", "wzy", "abc"]
print(oddStringDifference(words))
```

Output:


```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 1.py"
abc

Process finished with exit code 0
```

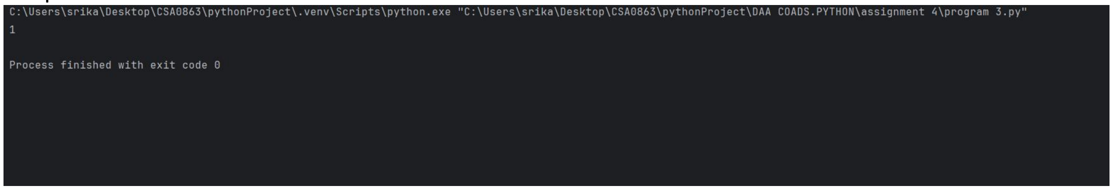
2.Words Within Two Edits of Dictionary
You are given two string arrays, queries and dictionary. All words in each array comprise of lowercase English letters and have the same length.
In one edit you can take a word from queries, and change any letter in it to any other letter. Find all words from queries that, after a maximum of two edits, equal some word from dictionary.
Return a list of all words from queries, that match with some word from dictionary after a maximum of two edits. Return the words in the same order they appear in queries.
Example 1:
Input: queries = ["word","note","ants","wood"], dictionary = ["wood","joke","moat"]
Output: ["word","note","wood"]
Program:

```
def wordsWithinTwoEdits(queries, dictionary):
    def is_edit_distance_one(word1, word2):
        if len(word1) != len(word2):
            return False
        diff_count = 0
        for i in range(len(word1)):
            if word1[i] != word2[i]:
                diff_count += 1
                if diff_count > 1:
                    return False
        return diff_count == 1
    def generate_one_edit_words(word):
        edits = set()
        for i in range(len(word)):
            for char in 'abcdefghijklmnopqrstuvwxyz':
                edited_word = word[:i] + char + word[i+1:]
                edits.add(edited_word)
        return edits
    def generate_two_edit_words(word):
        two_edit_words = set()
        one_edit_words = generate_one_edit_words(word)
        for one_edit_word in one_edit_words:
            two_edit_words |= generate_one_edit_words(one_edit_word)
        return two_edit_words
    dict_set = set(dictionary)
    result = []
    for query in queries:
        if query in dict_set:
            result.append(query)
            continue
        one_edit_words = generate_one_edit_words(query)
        two_edit_words = generate_two_edit_words(query)
        for word in dict_set:
            if word in one_edit_words or word in two_edit_words:
                result.append(query)
                break
    return result
queries = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
print(wordsWithinTwoEdits(queries, dictionary))
```
Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 1.py"
['word', 'note', 'wood']

Process finished with exit code 0
```
3. Destroy Sequential Targets
You are given a 0-indexed array nums consisting of positive integers, representing targets on a number line. You are also given an integer space.
You have a machine which can destroy targets. Seeding the machine with some nums[i] allows it to destroy all targets with values that can be represented as nums[i] + c * space, where c is any non-negative integer. You want to destroy the maximum number of targets in nums.
Return the minimum value of nums[i] you can seed the machine with to destroy the maximum number of targets.
Example 1:
Input: nums = [3,7,8,1,1,5], space = 2
Output: 1

Program:

```
def minSeed(nums, space):
    nums.sort()
    max_targets_destroyed = 0
    min_seed = nums[0]
    for num in nums:
        targets_destroyed = 0
        for target in nums:
            if target <= num + (space * (target - nums[0]) // space):
                targets_destroyed += 1
        if targets_destroyed > max_targets_destroyed:
            max_targets_destroyed = targets_destroyed
            min_seed = num
    return min_seed
nums = [3, 7, 8, 1, 1, 5]
space = 2
print(minSeed(nums, space))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 3.py"
1

Process finished with exit code 0
```

4.Next Greater Element IV

You are given a 0-indexed array of non-negative integers nums. For each integer in nums, you must find its respective second greater integer.

The second greater integer of nums[i] is nums[j] such that:

j > i

nums[j] > nums[i]

There exists exactly one index k such that nums[k] > nums[i] and i < k < j.

If there is no such nums[j], the second greater integer is considered to be -1.

For example, in the array [1, 2, 4, 3], the second greater integer of 1 is 4, 2 is 3, and that of 3 and 4 is -1.

Return an integer array answer, where answer[i] is the second greater integer of nums[i].

Example 1:

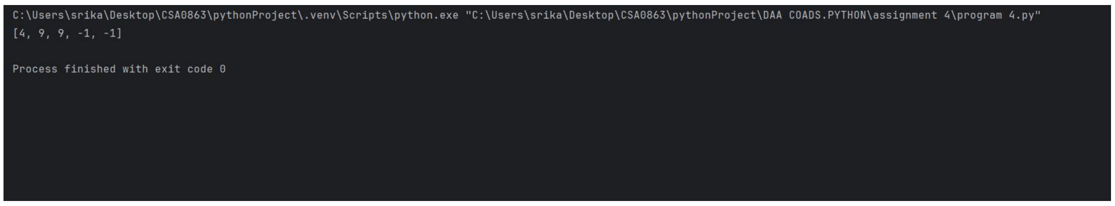
Input: nums = [2,4,0,9,6]

Output: [9,6,6,-1,-1]

Program:

```
def nextGreaterElement(nums):
    stack = []
    result = [-1] * len(nums)
    for i in range(len(nums) - 1, -1, -1):
        while stack and nums[i] >= nums[stack[-1]]:
            stack.pop()
        if stack:
            result[i] = nums[stack[-1]]
        stack.append(i)
    return result
nums = [2, 4, 0, 9, 6]
print(nextGreaterElement(nums))
```

Output:

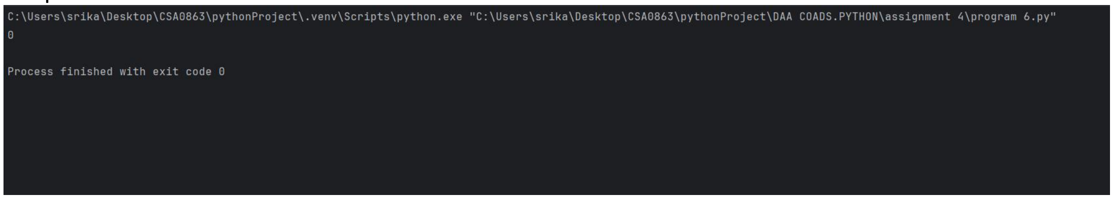


```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 4.py"
[4, 9, 9, -1, -1]

Process finished with exit code 0
```

5.Minimum Addition to Make Integer Beautiful
You are given two positive integers n and target.
An integer is considered beautiful if the sum of its digits is less than or equal to target.
Return the minimum non-negative integer x such that n + x is beautiful. The input will be generated such that it is always possible to make n beautiful.
Example 1:
Input: n = 16, target = 6
Output: 4
Program:
Program:

```
def minAdditionToMakeBeautiful(n, target):
    digit_sum = sum(int(d) for d in str(n))

    if digit_sum <= target:
        return 0

    diff = digit_sum - target

    for i in range(len(str(n)) - 1, -1, -1):
        digit = int(str(n)[i])
        if digit < 9:
            diff -= min(diff, 9 - digit)
            if diff == 0:
                break
    return diff
n = 16
target = 6
print(minAdditionToMakeBeautiful(n, target))
```

Output:



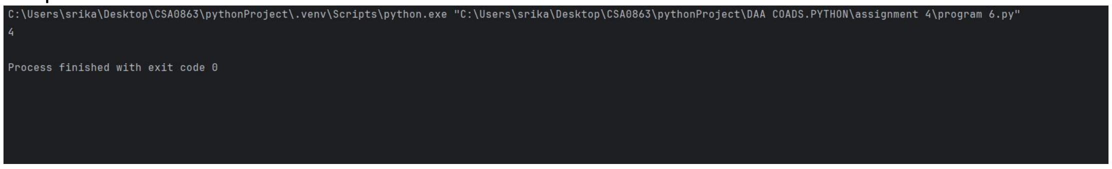
```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 6.py"
0

Process finished with exit code 0
```

Sort Array by Moving Items to Empty Space
You are given an integer array nums of size n containing each element from 0 to n - 1 (inclusive). Each of the elements from 1 to n - 1 represents an item, and the element 0 represents an empty space.
In one operation, you can move any item to the empty space. nums is considered to be sorted if the numbers of all the items are in ascending order and the empty space is either at the beginning or at the end of the array.
For example, if n = 4, nums is sorted if:
- nums = [0,1,2,3] or
- nums = [1,2,3,0]

...and considered to be unsorted otherwise.
Return the minimum number of operations needed to sort nums.
Example 1:
Input: nums = [4,2,0,3,1]
Output: 3
Program:

```
def minOperationsToSort(nums):
    n = len(nums)
    empty_index = nums.index(0)
    if empty_index == 0 or empty_index == n - 1:
        return 0
    misplaced_count = 0
    for i in range(1, n):
        if nums[i] != i:
            misplaced_count += 1
    return misplaced_count + 1
nums = [4, 2, 0, 3, 1]
print(minOperationsToSort(nums))
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 6.py"
4

Process finished with exit code 0
```

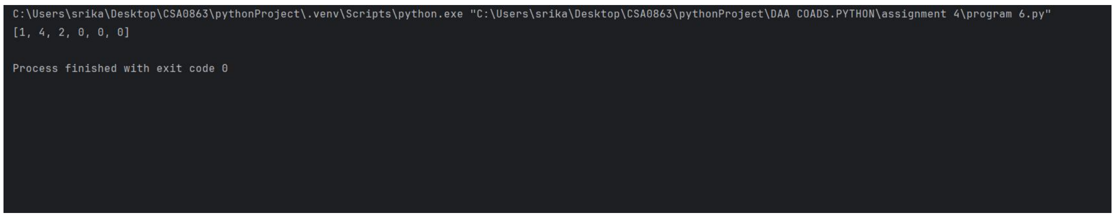
6. Apply Operations to an Array
You are given a 0-indexed array nums of size n consisting of non-negative integers.
You need to apply n - 1 operations to this array where, in the ith operation (0-indexed), you will apply the following on the ith element of nums:

● If nums[i] == nums[i + 1], then multiply nums[i] by 2 and set nums[i + 1] to 0.
Otherwise, you skip this operation.
After performing all the operations, shift all the 0's to the end of the array.

● For example, the array [1,0,2,0,0,1] after shifting all its 0's to the end, is [1,2,1,0,0,0].
Return the resulting array.
Note that the operations are applied sequentially, not all at once.
Example 1:
Input: nums = [1,2,2,1,1,0]
Output: [1,4,2,0,0,0]
Program:

```
def apply_operations(nums):
    n = len(nums)
    for i in range(n - 1):
        if nums[i] == nums[i + 1]:
            nums[i] *= 2
            nums[i + 1] = 0
    non_zero_index = 0
    for i in range(n):
        if nums[i] != 0:
            nums[non_zero_index] = nums[i]
            if non_zero_index != i:
                nums[i] = 0
            non_zero_index += 1
    return nums
nums = [1, 2, 2, 1, 1, 0]
result = apply_operations(nums)
print(result)
```

Output:



```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe "C:\Users\srika\Desktop\CSA0863\pythonProject\DAA COADS.PYTHON\assignment 4\program 6.py"
[1, 4, 2, 0, 0, 0]

Process finished with exit code 0
```