Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def odd_string_difference(words):
    def to_difference_array(word):
        return [ord(word[i+1]) - ord(word[i]) for i in range(len(word) - 1)]

    difference_arrays = [to_difference_array(word) for word in words]

    from collections import defaultdict
    diff_count = defaultdict(int)
    diff_index = defaultdict(list)

    for i, diff in enumerate(difference_arrays):
        diff_tuple = tuple(diff)
        diff_count[diff_tuple] += 1
        diff_index[diff_tuple].append(i)

    for diff, count in diff_count.items():
        if count == 1:
            unique_diff = diff
            break

    unique_index = diff_index[unique_diff][0]
    return words[unique_index]

words = ["adc", "wzy", "abc"]
print(odd_string_difference(words))
```

input

```
abc


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language: Python 3

main.py

```python
def words_within_two_edits(queries, dictionary):
    def can_transform_within_two_edits(word1, word2):

        differences = sum(1 for a, b in zip(word1, word2) if a != b)
        return differences <= 2

    result = []

    for query in queries:
        for dict_word in dictionary:
            if can_transform_within_two_edits(query, dict_word):
                result.append(query)
                break

    return result


queries = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
print(words_within_two_edits(queries, dictionary))

```

input

```
['word', 'note', 'wood']


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify

Language Python 3

main.py

```python
from collections import defaultdict

def destroy_max_targets(nums, space):
    remainder_count = defaultdict(int)
    remainder_targets = defaultdict(list)

    for num in nums:
        remainder = num % space
        remainder_count[remainder] += 1
        remainder_targets[remainder].append(num)

    max_count = max(remainder_count.values())
    candidates = [remainder for remainder, count in remainder_count.items() if count == max_count]

    best_remainder = min(candidates, key=lambda x: min(remainder_targets[x]))
    best_seed = min(remainder_targets[best_remainder])

    return best_seed

nums = [1, 3, 4, 9, 10]
space = 3
print(destroy_max_targets(nums, space))

```

input

```
1


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
def second_greater(nums):
    n = len(nums)
    result = [-1] * n
    stack = []
    greater_elements = {}

    for i in range(n - 1, -1, -1):
        while stack and nums[i] >= nums[stack[-1]]:
            idx = stack.pop()
            if idx not in greater_elements:
                greater_elements[idx] = []
            greater_elements[idx].append(nums[i])
            if len(greater_elements[idx]) == 2:
                result[idx] = greater_elements[idx][1]

        stack.append(i)

    return result

nums = [2, 4, 0, 9, 6]
print(second_greater(nums))
```

input

```
[-1, -1, -1, -1, -1]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | [] Beautify

Language: Python 3

main.py

```python
def average_value_of_even_numbers_divisible_by_three(nums):

    filtered_numbers = [num for num in nums if num % 2 == 0 and num % 3 == 0]

    if not filtered_numbers:
        return 0

    total_sum = sum(filtered_numbers)
    count = len(filtered_numbers)

    return total_sum // count

nums = [1, 3, 6, 9, 12, 15, 18]
print(average_value_of_even_numbers_divisible_by_three(nums))
```

input

```
12


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify

Language: Python 3

main.py

```python
def sum_of_digits(num):
    return sum(int(digit) for digit in str(num))

def find_min_x(n, target):
    x = 0
    while True:
        if sum_of_digits(n + x) <= target:
            return x
        x += 1

n = 58
target = 10
print(find_min_x(n, target))
```

input

```
2


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def sum_of_digits(num):
    return sum(int(digit) for digit in str(num))

def find_min_x(n, target):
    if sum_of_digits(n) <= target:
        return 0

    x = 0
    power_of_ten = 1
    while True:
        current_sum = sum_of_digits(n + x)
        if current_sum <= target:
            return x

        increment = power_of_ten - (n + x) % power_of_ten
        x += increment
        power_of_ten *= 10

n = 58
target = 10
print(find_min_x(n, target))
```

input

```
2


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
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

    result = []
    for num in nums:
        if num != 0:
            result.append(num)

    while len(result) < n:
        result.append(0)

    return result

nums = [2, 2, 3, 3, 0, 1]
print(apply_operations(nums))
```

input

```
[4, 6, 1, 0, 0, 0]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
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

    zero_count = nums.count(0)
    non_zero_nums = [num for num in nums if num != 0]
    return non_zero_nums + [0] * zero_count

nums1 = [1, 2, 2, 1, 1, 0]
nums2 = [0, 1]
print("Example 1:", apply_operations(nums1))
print("Example 2:", apply_operations(nums2))
```

input

```
Example 1: [1, 4, 2, 0, 0, 0]
Example 2: [1, 0]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def min_operations(nums):
    n = len(nums)
    count_from_start = 0
    count_from_end = 0

    # Count the number of items out of place from the beginning
    for i in range(n):
        if nums[i] != i:
            count_from_start += 1
        else:
            break

    # Count the number of items out of place from the end
    for i in range(n-1, -1, -1):
        if nums[i] != i:
            count_from_end += 1
        else:
            break

    return min(count_from_start, count_from_end)

# Test cases
print(min_operations([4,2,0,3,1]))  # Output: 3
print(min_operations([1,2,3,4,0]))  # Output: 0
print(min_operations([1,0,2,4,3]))  # Output: 2

```

input

```
1
5
2


...Program finished with exit code 0
Press ENTER to exit console.
```