1.Odd string difference

```
def odd_string_out(words):
    def get_diff_array(word):
        return [ord(word[i+1]) - ord(word[i]) for i in range(len(word) - 1)]
    diff_arrays = [get_diff_array(word) for word in words]
    for i in range(len(words)):
        if diff_arrays.count(diff_arrays[i]) == 1:
            return words[i]
    return None
print(odd_string_out(["adc","wzy","abc"]))
```

2.Words within two edits of dictionary

```
def words_within_two_edits(queries, dictionary):
    def is_within_two_edits(word1, word2):
        if len(word1) != len(word2):
            return False
        edits = sum(1 for a, b in zip(word1, word2) if a != b)
        return edits <= 2
    result = []
    for query in queries:
        if any(is_within_two_edits(query, word) for word in dictionary):
            result.append(query)
    return result
print(words_within_two_edits(["word","note","ants","wood"], ["wood","joke","moat"]))  # Output: 
["word", "note", "wood"]
```

3.Destroy sequential targets

```
def destroy_sequential_targets(nums, space):
    from collections import defaultdict
    count = defaultdict(int)
    for num in nums:
        count[num % space] += 1
    max_count = max(count.values())
    candidates = [num for num in nums if count[num % space] == max_count]
    return min(candidates)
print(destroy_sequential_targets([3,7,8,1,1,5], 2))
```

4.Minumum addition to make integer beautiful

```
def make_integer_beautiful(n, target):
    def digit_sum(x):
        return sum(int(d) for d in str(x))
    x = 0
    while digit_sum(n + x) > target:
        x += 1
    return x
print(make_integer_beautiful(16, 6))
```

5.Sort array by moving items to empty space

```
def sort_by_empty_space(nums):
    def find_zero(nums):
        return nums.index(0)
    n = len(nums)
    target = list(range(n))
    if nums == target or nums == target[::-1]:
        return 0
    moves = 0
    while nums != target:
        zero_index = find_zero(nums)
        if zero_index != 0:
            nums[zero_index], nums[nums[zero_index]] = nums[nums[zero_index]], nums[zero_index]
            moves += 1
        else:
```

```
        for i in range(1, n):
            if nums[i] != i:
                nums[0], nums[i] = nums[i], nums[0]
                moves += 1
                break
    return moves
print(sort_by_empty_space([4,2,0,3,1]))
```

6.Apply operations to an array

```
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 = [num for num in nums if num != 0] + [0] * nums.count(0)
    return result
print(apply_operations([1,2,2,1,1,0]))
```