MISTAKES

1. Self Dividing Numbers Code Mistakes:

- **Uninitialized variable num**:
  - num was used without being initialized, leading to undefined behavior.
- **Incorrect loop condition (while (left <= num && num <= right))**:
  - You were checking if num (which wasn't initialized) is within the range, causing logical errors. The loop should iterate from left to right, not involve num in the condition.
- **Incorrect use of pop_back()**:
  - Using pop_back() to remove elements from the result when digit == 0 is unnecessary and incorrect. You should simply skip the number if any digit is 0.
- **Incorrect modification of left and right inside the loop**:
  - Modifying both left++ and right-- inside the loop led to incorrect bounds for the iteration. This isn't needed and disrupts the logic for checking each number in the range.

2. String to Lowercase Code Mistakes:

- **Incorrect assignment of c in the loop**:
  - In your initial code, you tried to modify c inside the loop, but c was passed by value. Changing c didn't affect the string itself.
- **Returning the unchanged string**:
  - You didn't modify the string in-place; the string was never updated since c was a local variable, and you didn’t assign the modified character back to the string.
- **Lack of direct update to the string**:
  - You need to update the string’s characters by using a reference (char &c) so that the modifications reflect on the actual string.

3. Binary Search Code Mistakes:

- **Incorrect comparison in binary search**:
  - The condition if (target == mid) should be if (nums[mid] == target) because you're comparing the target with the value at mid in the array, not the index itself.
- **Logic for updating left and right**:
  - The logic for updating the left and right pointers in your binary search was a bit off. For example, you were using left++ and right--, but the usual binary search approach involves adjusting left = mid + 1 or right = mid - 1 based on comparisons with mid.

4. Palindrome Checking Code Mistakes:

- **Incorrect approach for removing characters**:
  - You tried to remove characters by modifying the string directly in a loop, but the correct approach for a palindrome check with removal involves checking if removing one character from either side can result in a valid palindrome.
- **Misuse of string methods (pop_back())**:
  - Using pop_back() was unnecessary and incorrect. You need to check if removing a character at the left or right would make the remaining string a palindrome, but you shouldn't directly modify the string in this way.

5. Invalid ASCII Value Code Mistakes:

- **Incorrect use of ASCII values**:
  - You were trying to convert uppercase to lowercase using ASCII values (A = 65, Z = 90), but you didn't correctly handle the string or character assignment for converting to lowercase.

6. Mistakes in Understanding and Using Functions:

- **Incorrect use of the while loop for self-dividing numbers**:
  - The while (left <= num && num <= right) condition was wrong. You need to iterate through the range from left to right, checking each number individually.
- **Incorrect division by zero check**:
  - In the code for checking self-dividing numbers, you didn't handle the case of division by zero properly, leading to runtime errors. You should check if a digit is 0 before performing the modulus operation.

General Mistakes Across Code:

- **Misunderstanding how to manipulate strings**:
  - In various codes, you had issues with modifying strings or characters in loops without using references or properly assigning modified values back to the string.
- **Not handling edge cases**:
  - In several problems (like the palindrome or self-dividing numbers), you didn't account for all edge cases such as when a digit is 0 or when the string/number is already valid.

Suggestions:

- **Use references (char &c)** when modifying characters in a string or vector.
- **Test conditions carefully** to ensure you're iterating over the correct range, checking the right values, and avoiding unnecessary operations like popping elements from vectors.
- **Update variables properly** when modifying ranges (left, right), using binary search logic, or processing digits in numbers.