Here are **100 C++ problem statements** that combine **`vector` STL** and **`<algorithm>` header functions** (like `sort`, `reverse`, `count`, `find`, `unique`, `accumulate`, `binary_search`, etc.). These are useful for practicing competitive programming and STL concepts:

◆ Basic Vector Operations (1–25)

1. Insert `n` integers into a vector and print them.
2. Find the sum of all elements in a vector.
3. Find the maximum element in a vector.
4. Find the minimum element in a vector.
5. Reverse a vector.
6. Sort a vector in ascending order.
7. Sort a vector in descending order.
8. Count the occurrences of a given number in a vector.
9. Check if a number exists in a vector using `find`.
10. Remove all occurrences of a given number.
11. Remove duplicate elements from a sorted vector.
12. Remove duplicate elements from an unsorted vector.
13. Get the unique elements using `set` and copy to vector.
14. Create a vector of squares of given `n` numbers.
15. Rotate a vector to the left by `k` positions.
16. Rotate a vector to the right by `k` positions.
17. Check if the vector is sorted in ascending order.
18. Check if the vector is sorted in descending order.
19. Check if a vector is palindrome.
20. Merge two sorted vectors into one.
21. Find the second largest element in a vector.
22. Find the third smallest element in a vector.
23. Count even and odd numbers in a vector.
24. Partition the vector into even and odd numbers.
25. Remove all zeros from a vector.

◆ Using Standard Algorithms (26–50)

26. Use `accumulate` to calculate average of elements.
27. Use `all_of` to check if all elements are positive.
28. Use `any_of` to check if any element is negative.
29. Use `none_of` to check if no element is zero.
30. Use `is_sorted` to check if the vector is sorted.
31. Use `equal` to check if two vectors are the same.
32. Use `binary_search` to check for existence of an element.
33. Use `lower_bound` to find first element not less than `x`.
34. Use `upper_bound` to find first element greater than `x`.
35. Use `iota` to fill a vector with increasing numbers.
36. Use `next_permutation` to find next lexicographic permutation.

37. Use `prev_permutation` for previous permutation.
38. Find first element greater than average using `find_if`.
39. Count elements divisible by 3 using `count_if`.
40. Find all prime numbers in a vector using `remove_if`.
41. Replace all negative values with 0 using `replace_if`.
42. Use `reverse` with `vector::begin()` and `end()`.
43. Use `max_element` to find index of max value.
44. Use `min_element` to find index of min value.
45. Calculate product of all elements using `accumulate`.
46. Use `fill` to set all values to a constant.
47. Use `generate` to fill with random numbers.
48. Use `inplace_merge` on two halves of sorted vector.
49. Use `partition` to segregate even and odd numbers.
50. Use `stable_partition` to separate positive and negative numbers.

◆ Intermediate Level (51–75)

51. Find subarray with maximum sum (Kadane's using vector).
52. Find the most frequent element.
53. Find the median of a vector.
54. Find the mode of a vector.
55. Print all elements that appear exactly once.
56. Sort vector of pairs by second value.
57. Sort vector of pairs by first descending, second ascending.
58. Count pairs with sum equal to `k`.
59. Find the longest increasing subarray.
60. Find the longest subarray with equal even and odd count.
61. Find the longest contiguous subarray with sum `k`.
62. Rotate a vector by 1 place using STL.
63. Create a vector of factorials up to `n`.
64. Remove elements greater than a given value.
65. Compress values to ranks using sorting and mapping.
66. Merge intervals stored as vector of pairs.
67. Implement frequency sort using vector and map.
68. Remove duplicate strings from vector using `sort + unique`.
69. Print all elements greater than previous element.
70. Count number of elements between two values using `lower_bound` and `upper_bound`.
71. Print top `k` largest elements using `nth_element`.
72. Replace even elements with square.
73. Replace odd elements with cube.
74. Find longest subarray with alternating signs.
75. Use `for_each` to print squares of elements.

◆ Advanced Logic (76–100)

76. Rearrange vector so that even and odd alternate.
77. Find duplicate elements in an unsorted vector.
78. Find missing number from 1 to n using `accumulate`.
79. Implement sieve of Eratosthenes using vector.
80. Find leaders in an array (right greater elements).
81. Rearrange elements in zig-zag fashion.
82. Shuffle vector randomly using `random_shuffle` or `shuffle`.
83. Check if two vectors are permutations of each other.
84. Find intersection of two vectors.
85. Find union of two vectors using `set_union`.
86. Rotate vector until it becomes sorted.
87. Count inversions in array using modified merge sort.
88. Find maximum product subarray.
89. Calculate span of stock prices.
90. Rearrange to form largest number using `to_string` and sort.
91. Count elements that are greater than all elements to their right.
92. Count peaks in the vector (i.e., `v[i] > v[i-1] and v[i] > v[i+1]`)
93. Compute prefix sum array using `partial_sum`.
94. Compute suffix sum array.
95. Detect if vector is rotated sorted array.
96. Find index of equilibrium point (left sum = right sum).
97. Print subarrays with 0 sum using hashing.
98. Count triplets with given sum.
99. Rotate vector in-place using `rotate` STL.
100. Check if array is bitonic (first increasing then decreasing).