Challenge Set 1 (1–10)

---

1. Filter Even Numbers

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5, 6);
numbers.stream()
       .filter(n -> n % 2 == 0)
       .forEach(System.out::println); // Output: 2, 4, 6
```

---

2. Convert Names to Uppercase

```
List<String> names = List.of("alice", "bob", "charlie");
List<String> upper = names.stream()
                          .map(String::toUpperCase)
                          .collect(Collectors.toList());
System.out.println(upper); // Output: [ALICE, BOB, CHARLIE]
```

---

3. Find First City Starting With 'P'

```
List<String> cities = List.of("Mumbai", "Paris", "Pune");
cities.stream()
      .filter(c -> c.startsWith("P"))
      .findFirst()
      .ifPresent(System.out::println); // Output: Paris
```

---

4. Filter Numbers Divisible by 5 But Not 10

```
List<Integer> nums = List.of(5, 10, 15, 20, 25);
nums.stream()
    .filter(n -> n % 5 == 0)
    .filter(n -> n % 10 != 0)
    .forEach(System.out::println); // Output: 5, 15, 25
```

---

5. Flatten a Nested List

```
List<List<String>> nested = List.of(List.of("a", "b"), List.of("c", "d"));
nested.stream()
      .flatMap(List::stream)
      .forEach(System.out::println); // Output: a b c d
```

---

6. Get Length of Each Fruit

```
List<String> fruits = List.of("apple", "banana", "cherry");
List<Integer> lengths = fruits.stream()
                             .map(String::length)
                             .collect(Collectors.toList());
System.out.println(lengths); // Output: [5, 6, 6]
```

---

7. Filter Names of Length 5 and Start with A

```
List<String> names = List.of("Ankit", "Aryan", "Alice", "John");
List<String> result = names.stream()
                           .filter(n -> n.length() == 5)
                           .filter(n -> n.startsWith("A"))
                           .collect(Collectors.toList());
System.out.println(result); // Output: [Ankit, Alice]
```

---

8. Group Animals by Length

```
List<String> animals = List.of("cat", "elephant", "dog", "lion");
Map<Integer, List<String>> grouped = animals.stream()
    .collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {3=[cat, dog], 4=[lion], 8=[elephant]}
```

---

9. Sum of Scores

```
List<Integer> scores = List.of(10, 20, 30);
int sum = scores.stream()
                .mapToInt(Integer::intValue)
                .sum();
System.out.println(sum); // Output: 60
```

---

10. Flatten and Sort

```
List<List<Integer>> nested = List.of(List.of(5, 1), List.of(3, 2));
List<Integer> flatSorted = nested.stream()
    .flatMap(List::stream)
    .sorted()
    .collect(Collectors.toList());
System.out.println(flatSorted); // Output: [1, 2, 3, 5]
```

Challenge Set 2 (11–20)

---

11. Partition Numbers by Even/Odd

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
Map<Boolean, List<Integer>> partitioned = numbers.stream()
    .collect(Collectors.partitioningBy(n -> n % 2 == 0));
System.out.println(partitioned);
// Output: {false=[1, 3, 5], true=[2, 4]}
```

---

12. Group Words by First Character

```
List<String> words = List.of("apple", "banana", "apricot", "blueberry");
Map<Character, List<String>> grouped = words.stream()
    .collect(Collectors.groupingBy(w -> w.charAt(0)));
System.out.println(grouped);
// Output: {a=[apple, apricot], b=[banana, blueberry]}
```

---

13. Reduce to Product of Numbers

```
List<Integer> nums = List.of(1, 2, 3, 4);
int product = nums.stream()
                  .reduce(1, (acc, n) -> acc * n);
System.out.println(product); // Output: 24
```

---

14. Concatenate Words with Comma

```
List<String> words = List.of("", "Python", "C++");
String result = words.stream()
                     .reduce("", (a, b) -> a.isEmpty() ? b : a + ", " + b);
System.out.println(result); // Output: , Python, C++
```

---

15. Find Longest Word Using Reduce

```
List<String> words = List.of("apple", "banana", "cherry");
String longest = words.stream()
                      .reduce((w1, w2) -> w1.length() > w2.length() ? w1 :
w2)
                      .orElse("");
System.out.println(longest); // Output: banana
```

---

16. Check If Any Number is Negative

```
List<Integer> nums = List.of(1, -2, 3, 4);
boolean hasNegative = nums.stream()
                          .anyMatch(n -> n < 0);
System.out.println(hasNegative); // Output: true
```

---

17. Collect Words Starting with 'M' into a List

```
List<String> words = List.of("Mike", "Mary", "John", "Mark");
List<String> mWords = words.stream()
                           .filter(w -> w.startsWith("M"))
                           .collect(Collectors.toList());
System.out.println(mWords); // Output: [Mike, Mary, Mark]
```

---

18. Map Words to Their Lengths

```
List<String> words = List.of("", "Python");
Map<String, Integer> map = words.stream()
    .collect(Collectors.toMap(w -> w, String::length));
System.out.println(map);
// Output: {=4, Python=6}
```

---

19. Count Frequency of Each Word

```
List<String> words = List.of("apple", "banana", "apple");
Map<String, Long> freq = words.stream()
    .collect(Collectors.groupingBy(w -> w, Collectors.counting()));
System.out.println(freq);
// Output: {apple=2, banana=1}
```

---

20. Convert List to Set

```
List<String> list = List.of("a", "b", "a", "c");
Set<String> set = list.stream()
                      .collect(Collectors.toSet());
System.out.println(set); // Output: [a, b, c]
```

Challenge Set 3 (21–30)

---

21. Find Max Number Using Reduce

```
List<Integer> nums = List.of(3, 5, 2, 10);
int max = nums.stream()
              .reduce(Integer.MIN_VALUE, Integer::max);
System.out.println(max); // Output: 10
```

---

22. Filter and Sum Even Numbers

```
List<Integer> nums = List.of(1, 2, 3, 4, 5);
int sumEven = nums.stream()
                  .filter(n -> n % 2 == 0)
                  .mapToInt(Integer::intValue)
                  .sum();
System.out.println(sumEven); // Output: 6
```

---

23. Convert Strings to Lowercase

```
List<String> words = List.of("", "PYTHON");
List<String> lower = words.stream()
                          .map(String::toLowerCase)
                          .collect(Collectors.toList());
System.out.println(lower); // Output: [, python]
```

---

24. Find Any Word Starting with ‘J’

```
List<String> words = List.of("", "Python", "Script");
Optional<String> anyJ = words.stream()
                             .filter(w -> w.startsWith("J"))
                             .findAny();
anyJ.ifPresent(System.out::println); // Output:  (or Script)
```

---

25. Count Number of Empty Strings

```
List<String> list = List.of("abc", "", "def", "", "");
long countEmpty = list.stream()
                      .filter(String::isEmpty)
                      .count();
System.out.println(countEmpty); // Output: 3
```

---

26. Remove Duplicates Using Distinct

```
List<String> list = List.of("a", "b", "a", "c");
List<String> distinct = list.stream()
                            .distinct()
                            .collect(Collectors.toList());
System.out.println(distinct); // Output: [a, b, c]
```

---

27. Convert IntStream to List<Integer>

```
List<Integer> list = IntStream.range(1, 5)
                              .boxed()
                              .collect(Collectors.toList());
System.out.println(list); // Output: [1, 2, 3, 4]
```

---

28. Flatten Nested Lists

```
List<List<String>> nested = List.of(
    List.of("a", "b"),
    List.of("c", "d")
);
List<String> flat = nested.stream()
                          .flatMap(List::stream)
                          .collect(Collectors.toList());
System.out.println(flat); // Output: [a, b, c, d]
```

---

29. Sort List of Strings by Length

```
List<String> words = List.of("apple", "banana", "fig");
List<String> sorted = words.stream()
                           .sorted(Comparator.comparingInt(String::length))
                           .collect(Collectors.toList());
System.out.println(sorted); // Output: [fig, apple, banana]
```

---

30. Collect to LinkedHashSet to Preserve Order

```
List<String> list = List.of("a", "b", "a", "c");
LinkedHashSet<String> linkedSet = list.stream()

.collect(Collectors.toCollection(LinkedHashSet::new));
System.out.println(linkedSet); // Output: [a, b, c]
```

Challenge Set 4 (31–40)

---

31. Check if All Strings Are Non-Empty

```
List<String> list = List.of("hello", "world", "");
boolean allNonEmpty = list.stream()
                          .allMatch(s -> !s.isEmpty());
System.out.println(allNonEmpty); // Output: false
```

---

32. Check if Any String Starts with "A"

```
List<String> names = List.of("Alice", "Bob", "Charlie");
boolean anyStartsWithA = names.stream()
                              .anyMatch(name -> name.startsWith("A"));
System.out.println(anyStartsWithA); // Output: true
```

---

33. Skip First 3 Elements

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5, 6);
List<Integer> skipped = numbers.stream()
                               .skip(3)
                               .collect(Collectors.toList());
System.out.println(skipped); // Output: [4, 5, 6]
```

---

34. Limit Stream to First 2 Elements

```
List<String> words = List.of("one", "two", "three");
List<String> limited = words.stream()
                            .limit(2)
                            .collect(Collectors.toList());
System.out.println(limited); // Output: [one, two]
```

---

35. Sum of Lengths of All Strings

```
List<String> words = List.of("cat", "dog", "elephant");
int totalLength = words.stream()
                       .mapToInt(String::length)
                       .sum();
System.out.println(totalLength); // Output: 14
```

---

36. Find Longest String Using Reduce

```
List<String> words = List.of("a", "ab", "abc");
Optional<String> longest = words.stream()
                                .reduce((w1, w2) -> w1.length() >
w2.length() ? w1 : w2);
longest.ifPresent(System.out::println); // Output: abc
```

---

37. Collect to TreeSet (Sorted & Unique)

```
List<String> list = List.of("banana", "apple", "banana");
TreeSet<String> treeSet = list.stream()

.collect(Collectors.toCollection(TreeSet::new));
System.out.println(treeSet); // Output: [apple, banana]
```

---

38. Group Strings by Length

```
List<String> words = List.of("a", "to", "tea", "ted");
Map<Integer, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {1=[a], 2=[to], 3=[tea, ted]}
```

---

39. Partition Numbers Into Even and Odd

```
List<Integer> nums = List.of(1, 2, 3, 4, 5);
Map<Boolean, List<Integer>> partitioned = nums.stream()

.collect(Collectors.partitioningBy(n -> n % 2 == 0));
System.out.println(partitioned);
// Output: {false=[1, 3, 5], true=[2, 4]}
```

---

40. Map Strings to Their Frequencies

```
List<String> words = List.of("apple", "banana", "apple", "orange",
"banana");
Map<String, Long> frequency = words.stream()
```

```
.collect(Collectors.groupingBy(Function.identity(),
Collectors.counting()));
System.out.println(frequency);
// Output: {orange=1, banana=2, apple=2}
```

Challenge Set 5 (41–50)

---

41. Find the Minimum Integer in a Stream

```
List<Integer> numbers = List.of(5, 3, 9, 1, 6);
Optional<Integer> min = numbers.stream().min(Integer::compareTo);
min.ifPresent(System.out::println); // Output: 1
```

---

42. Find the Maximum Integer in a Stream

```
List<Integer> numbers = List.of(5, 3, 9, 1, 6);
Optional<Integer> max = numbers.stream().max(Integer::compareTo);
max.ifPresent(System.out::println); // Output: 9
```

---

43. Convert Stream to Array

```
List<String> words = List.of("a", "b", "c");
String[] array = words.stream().toArray(String[]::new);
System.out.println(Arrays.toString(array)); // Output: [a, b, c]
```

---

44. Find the First Element Matching a Condition

```
List<String> names = List.of("John", "Jane", "Adam", "Alice");
Optional<String> startsWithA = names.stream()
                                    .filter(name -> name.startsWith("A"))
                                    .findFirst();
startsWithA.ifPresent(System.out::println); // Output: Adam
```

---

45. Collect Distinct Elements

```
List<Integer> nums = List.of(1, 2, 2, 3, 4, 4);
```

```
List<Integer> distinct =
nums.stream().distinct().collect(Collectors.toList());
System.out.println(distinct); // Output: [1, 2, 3, 4]
```

---

46. Generate an Infinite Stream and Limit It

```
Stream<Integer> infinite = Stream.iterate(1, n -> n + 1);
List<Integer> firstFive = infinite.limit(5).collect(Collectors.toList());
System.out.println(firstFive); // Output: [1, 2, 3, 4, 5]
```

---

47. Filter Null Values From a Stream

```
List<String> list = List.of("a", null, "b", null, "c");
List<String> filtered = list.stream()
                            .filter(Objects::nonNull)
                            .collect(Collectors.toList());
System.out.println(filtered); // Output: [a, b, c]
```

---

48. Use peek() for Debugging Stream Pipeline

```
List<String> names = List.of("John", "Jane", "Adam");
List<String> result = names.stream()
                           .peek(n -> System.out.println("Before filter: "
+ n))
                           .filter(n -> n.startsWith("J"))
                           .peek(n -> System.out.println("After filter: " +
n))
                           .collect(Collectors.toList());
// Output includes intermediate peek prints
```

---

49. Summarize Statistics of Integers

```
List<Integer> nums = List.of(3, 4, 7, 1, 9);
IntSummaryStatistics stats =
nums.stream().mapToInt(Integer::intValue).summaryStatistics();
System.out.println(stats);
// Output: IntSummaryStatistics{count=5, sum=24, min=1, average=4.8, max=9}
```

---

50. Convert Stream of Strings to CSV Format

```
List<String> words = List.of("apple", "banana", "cherry");
```

```
String csv = words.stream().collect(Collectors.joining(", "));
System.out.println(csv); // Output: apple, banana, cherry
```

Challenge Set 6 (51–60)

---

51. Count the Frequency of Each Element in a List

```
List<String> words = List.of("apple", "banana", "apple", "cherry",
"banana", "apple");
Map<String, Long> frequency = words.stream()
                                    .collect(Collectors.groupingBy(w -> w,
Collectors.counting()));
System.out.println(frequency);
// Output: {banana=2, cherry=1, apple=3}
```

---

52. Group Objects by a Property

```
class Person {
    String name;
    int age;
    Person(String n, int a) { name = n; age = a; }
    public int getAge() { return age; }
}
List<Person> people = List.of(new Person("Alice", 20), new Person("Bob",
30), new Person("Carol", 20));
Map<Integer, List<Person>> groupedByAge = people.stream()

.collect(Collectors.groupingBy(Person::getAge));
System.out.println(groupedByAge);
// Output: {20=[Alice, Carol], 30=[Bob]}
```

---

53. Create a Map with Key as Length and Value as List of Strings of That Length

```
List<String> words = List.of("apple", "bat", "car", "banana", "cat");
Map<Integer, List<String>> groupedByLength = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(groupedByLength);
// Output: {3=[bat, car, cat], 5=[apple], 6=[banana]}
```

---

54. Partition a List into Even and Odd Numbers

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5, 6);
Map<Boolean, List<Integer>> partitioned = numbers.stream()

.collect(Collectors.partitioningBy(n -> n % 2 == 0));
System.out.println(partitioned);
// Output: {false=[1, 3, 5], true=[2, 4, 6]}
```

---

55. Concatenate Strings Using reduce()

```
List<String> words = List.of("", "is", "fun");
String sentence = words.stream()
                       .reduce("", (a, b) -> a.isEmpty() ? b : a + " " +
b);
System.out.println(sentence); // Output:  is fun
```

---

56. Use mapToInt() to Calculate Sum of String Lengths

```
List<String> words = List.of("apple", "banana", "cherry");
int totalLength = words.stream()
                       .mapToInt(String::length)
                       .sum();
System.out.println(totalLength); // Output: 17
```

---

57. Find Average of Numbers Using mapToDouble()

```
List<Integer> numbers = List.of(10, 20, 30, 40);
double avg = numbers.stream()
                    .mapToDouble(Integer::doubleValue)
                    .average()
                    .orElse(0);
System.out.println(avg); // Output: 25.0
```

---

58. Convert IntStream to Stream<Integer>

```
List<Integer> list = IntStream.range(1, 5)
                              .boxed()
                              .collect(Collectors.toList());
System.out.println(list); // Output: [1, 2, 3, 4]
```

---

59. Use reduce() to Find Longest String

```
List<String> words = List.of("apple", "banana", "cherry", "date");
Optional<String> longest = words.stream()
                                .reduce((w1, w2) -> w1.length() >=
w2.length() ? w1 : w2);
longest.ifPresent(System.out::println); // Output: banana
```

---

60. Count Distinct Characters in a String

```
String word = "hello";
long distinctChars = word.chars()
                         .distinct()
                         .count();
System.out.println(distinctChars); // Output: 4 (h, e, l, o)
```

Challenge Set 7 (61–70)

---

61. Filter out Null Elements and Collect Non-null Values

```
List<String> list = Arrays.asList("apple", null, "banana", null, "cherry");
List<String> filtered = list.stream()
                            .filter(Objects::nonNull)
                            .collect(Collectors.toList());
System.out.println(filtered); // Output: [apple, banana, cherry]
```

---

62. Flatten a List of Lists into a Single List

```
List<List<String>> nestedList = List.of(
    List.of("a", "b"),
    List.of("c", "d", "e")
);
List<String> flatList = nestedList.stream()
                                  .flatMap(Collection::stream)
                                  .collect(Collectors.toList());
System.out.println(flatList); // Output: [a, b, c, d, e]
```

---

63. Sort a List of Objects by a Field Using Comparator

```
class Person {
    String name;
    int age;
    Person(String n, int a) { name = n; age = a; }
    public int getAge() { return age; }
```

```
}
List<Person> people = List.of(new Person("Alice", 25), new Person("Bob",
20));
List<Person> sortedByAge = people.stream()

.sorted(Comparator.comparingInt(Person::getAge))
                                  .collect(Collectors.toList());
sortedByAge.forEach(p -> System.out.println(p.name + " " + p.age));
// Output: Bob 20
//         Alice 25
```

---

64. Find Maximum Value Using reduce()

```
List<Integer> numbers = List.of(5, 10, 3, 7);
int max = numbers.stream()
                 .reduce(Integer::max)
                 .orElse(-1);
System.out.println(max); // Output: 10
```

---

65. Generate a Stream of Random Numbers and Limit to 5

```
new Random().ints()
            .limit(5)
            .forEach(System.out::println);
```

---

66. Create a Map from a List Using toMap()

```
List<String> words = List.of("apple", "banana", "cherry");
Map<String, Integer> map = words.stream()
                                .collect(Collectors.toMap(w -> w,
String::length));
System.out.println(map);
// Output: {apple=5, banana=6, cherry=6}
```

---

67. Join Strings Using Collectors.joining()

```
List<String> words = List.of("", "Stream", "API");
String joined = words.stream()
                     .collect(Collectors.joining(", "));
System.out.println(joined); // Output: , Stream, API
```

---

68. Create an IntSummaryStatistics Object

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
IntSummaryStatistics stats = numbers.stream()
                                    .mapToInt(Integer::intValue)
                                    .summaryStatistics();
System.out.println("Max: " + stats.getMax());
System.out.println("Min: " + stats.getMin());
System.out.println("Sum: " + stats.getSum());
System.out.println("Avg: " + stats.getAverage());
```

---

69. Filter Strings Starting with a Specific Letter

```
List<String> words = List.of("apple", "banana", "avocado", "cherry");
List<String> filtered = words.stream()
                             .filter(w -> w.startsWith("a"))
                             .collect(Collectors.toList());
System.out.println(filtered); // Output: [apple, avocado]
```

---

70. Use peek() to Debug Stream Pipeline

```
List<String> words = List.of("one", "two", "three");
List<String> result = words.stream()
                           .peek(w -> System.out.println("Original: " + w))
                           .map(String::toUpperCase)
                           .peek(w -> System.out.println("Uppercase: " +
w))
                           .collect(Collectors.toList());
System.out.println(result);
// Output includes debug prints and final list
```

Challenge Set 8 (71–80)

---

71. Convert a Stream to an Array

```
List<String> fruits = List.of("apple", "banana", "cherry");
String[] arr = fruits.stream()
                     .toArray(String[]::new);
System.out.println(Arrays.toString(arr)); // Output: [apple, banana,
cherry]
```

---

72. Remove Duplicates Using distinct()

```
List<Integer> numbers = List.of(1, 2, 2, 3, 3, 3);
List<Integer> distinctNumbers = numbers.stream()
                                       .distinct()
                                       .collect(Collectors.toList());
System.out.println(distinctNumbers); // Output: [1, 2, 3]
```

---

73. Find Any Element Matching a Condition

```
List<String> names = List.of("Alice", "Bob", "Charlie");
Optional<String> anyStartingWithB = names.stream()
                                         .filter(n -> n.startsWith("B"))
                                         .findAny();
anyStartingWithB.ifPresent(System.out::println); // Output: Bob
```

---

74. Check if All Elements Match a Predicate

```
List<Integer> numbers = List.of(2, 4, 6, 8);
boolean allEven = numbers.stream()
                         .allMatch(n -> n % 2 == 0);
System.out.println(allEven); // Output: true
```

---

75. Check if Any Element Matches a Predicate

```
List<Integer> numbers = List.of(1, 3, 5, 6);
boolean anyEven = numbers.stream()
                         .anyMatch(n -> n % 2 == 0);
System.out.println(anyEven); // Output: true
```

---

76. Collect to a Set

```
List<String> words = List.of("apple", "banana", "apple");
Set<String> set = words.stream()
                       .collect(Collectors.toSet());
System.out.println(set); // Output: [banana, apple]
```

---

77. Skip First N Elements

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
List<Integer> skipped = numbers.stream()
                               .skip(2)
```

```
                                .collect(Collectors.toList());
System.out.println(skipped); // Output: [3, 4, 5]
```

---

78. Limit Stream to First N Elements

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
List<Integer> limited = numbers.stream()
                                .limit(3)
                                .collect(Collectors.toList());
System.out.println(limited); // Output: [1, 2, 3]
```

---

79. Generate a Stream of Squares

```
List<Integer> numbers = List.of(1, 2, 3, 4);
List<Integer> squares = numbers.stream()
                                .map(n -> n * n)
                                .collect(Collectors.toList());
System.out.println(squares); // Output: [1, 4, 9, 16]
```

---

80. Group by Length of Strings

```
List<String> words = List.of("apple", "bat", "car", "door");
Map<Integer, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {3=[bat, car], 4=[door], 5=[apple]}
```

Challenge Set 9 (81–90)

---

81. Find the Longest String

```
List<String> words = List.of("cat", "house", "elephant", "dog");
Optional<String> longest = words.stream()
                                 .reduce((w1, w2) -> w1.length() >
w2.length() ? w1 : w2);
longest.ifPresent(System.out::println);  // Output: elephant
```

---

82. Calculate Average of Integers

```
List<Integer> numbers = List.of(10, 20, 30, 40);
double average = numbers.stream()
                        .mapToInt(Integer::intValue)
                        .average()
                        .orElse(0);
System.out.println(average);  // Output: 25.0
```

---

83. Partition Numbers into Even and Odd

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5, 6);
Map<Boolean, List<Integer>> partitioned = numbers.stream()

.collect(Collectors.partitioningBy(n -> n % 2 == 0));
System.out.println(partitioned);
// Output: {false=[1, 3, 5], true=[2, 4, 6]}
```

---

84. Find Maximum Value

```
List<Integer> numbers = List.of(5, 12, 7, 9);
Optional<Integer> max = numbers.stream()
                               .max(Integer::compare);
max.ifPresent(System.out::println);  // Output: 12
```

---

85. Count Words Starting With a Given Letter

```
List<String> words = List.of("apple", "apricot", "banana", "avocado");
long count = words.stream()
                  .filter(w -> w.startsWith("a"))
                  .count();
System.out.println(count);  // Output: 3
```

---

86. Join Strings with a Delimiter

```
List<String> words = List.of("", "Python", "C++");
String result = words.stream()
                     .collect(Collectors.joining(", "));
System.out.println(result);  // Output: , Python, C++
```

---

87. Sum of Squares

```
List<Integer> numbers = List.of(1, 2, 3, 4);
int sumSquares = numbers.stream()
                        .map(n -> n * n)
                        .reduce(0, Integer::sum);
System.out.println(sumSquares);  // Output: 30
```

---

88. Find First Element Matching a Condition

```
List<String> words = List.of("apple", "banana", "cherry");
Optional<String> firstStartingWithB = words.stream()
                                           .filter(w -> w.startsWith("b"))
                                           .findFirst();
firstStartingWithB.ifPresent(System.out::println);  // Output: banana
```

---

89. Convert Stream to Map (word -> length)

```
List<String> words = List.of("one", "three", "five");
Map<String, Integer> map = words.stream()
                                .collect(Collectors.toMap(w -> w,
String::length));
System.out.println(map);
// Output: {one=3, three=5, five=4}
```

---

90. Check if Stream Contains Any Nulls

```
List<String> list = List.of("a", null, "c");
boolean hasNull = list.stream()
                      .anyMatch(Objects::isNull);
System.out.println(hasNull);  // Output: true
```

Challenge Set 10 (91–100)

---

91. Filter Strings Containing a Substring

```
List<String> words = List.of("apple", "application", "banana", "apply");
List<String> filtered = words.stream()
                             .filter(w -> w.contains("app"))
                             .collect(Collectors.toList());
System.out.println(filtered);  // Output: [apple, application, apply]
```

---

92. Sum of Odd Numbers

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
int sumOdds = numbers.stream()
                     .filter(n -> n % 2 != 0)
                     .mapToInt(Integer::intValue)
                     .sum();
System.out.println(sumOdds);  // Output: 9
```

---

93. Collect Unique Characters from Words

```
List<String> words = List.of("apple", "banana");
Set<Character> uniqueChars = words.stream()
                                  .flatMapToInt(String::chars)
                                  .mapToObj(c -> (char) c)
                                  .collect(Collectors.toSet());
System.out.println(uniqueChars);  // Output: [a, b, e, l, n, p]
```

---

94. Find Minimum Value

```
List<Integer> numbers = List.of(7, 3, 9, 1);
Optional<Integer> min = numbers.stream()
                               .min(Integer::compare);
min.ifPresent(System.out::println);  // Output: 1
```

---

95. Group Strings by Length

```
List<String> words = List.of("cat", "dog", "elephant", "bee");
Map<Integer, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {3=[cat, dog, bee], 8=[elephant]}
```

---

96. Count Total Characters in All Strings

```
List<String> words = List.of("hello", "world");
int totalChars = words.stream()
                      .mapToInt(String::length)
                      .sum();
System.out.println(totalChars);  // Output: 10
```

---

97. Check if All Numbers Are Positive

```
List<Integer> numbers = List.of(1, 2, 3, 4);
boolean allPositive = numbers.stream()
                             .allMatch(n -> n > 0);
System.out.println(allPositive);  // Output: true
```

---

98. Find Any Even Number

```
List<Integer> numbers = List.of(1, 3, 5, 6, 7);
Optional<Integer> anyEven = numbers.stream()
                                   .filter(n -> n % 2 == 0)
                                   .findAny();
anyEven.ifPresent(System.out::println);  // Output: 6
```

---

99. Convert List of Integers to Comma-Separated String

```
List<Integer> numbers = List.of(1, 2, 3);
String csv = numbers.stream()
                    .map(String::valueOf)
                    .collect(Collectors.joining(", "));
System.out.println(csv);  // Output: 1, 2, 3
```

---

100. Create Map of Word to Frequency

```
List<String> words = List.of("apple", "banana", "apple", "banana",
"banana");
Map<String, Long> frequency = words.stream()
                                   .collect(Collectors.groupingBy(w -> w,
Collectors.counting()));
System.out.println(frequency);
// Output: {apple=2, banana=3}
```

Challenge Set 11 (101–110)

---

101. Find Longest Word

```
List<String> words = List.of("cat", "elephant", "dog");
Optional<String> longest = words.stream()
```

```
.max(Comparator.comparingInt(String::length));
longest.ifPresent(System.out::println);  // Output: elephant
```

---

102. Sort Numbers in Descending Order

```
List<Integer> numbers = List.of(3, 1, 4, 1, 5);
List<Integer> sortedDesc = numbers.stream()
                                  .sorted(Comparator.reverseOrder())
                                  .collect(Collectors.toList());
System.out.println(sortedDesc);  // Output: [5, 4, 3, 1, 1]
```

---

103. Get Distinct Squares of Numbers

```
List<Integer> numbers = List.of(2, -2, 3, 3);
List<Integer> squares = numbers.stream()
                               .map(n -> n * n)
                               .distinct()
                               .collect(Collectors.toList());
System.out.println(squares);  // Output: [4, 9]
```

---

104. Find First String Starting With Letter

```
List<String> words = List.of("apple", "banana", "cherry");
Optional<String> firstB = words.stream()
                               .filter(w -> w.startsWith("b"))
                               .findFirst();
firstB.ifPresent(System.out::println);  // Output: banana
```

---

105. Count Words Longer Than N

```
List<String> words = List.of("", "stream", "lambda", "code");
long count = words.stream()
                  .filter(w -> w.length() > 4)
                  .count();
System.out.println(count);  // Output: 2
```

---

106. Create a List of Lengths of Strings

```
List<String> words = List.of("", "stream");
List<Integer> lengths = words.stream()
                             .map(String::length)
```

```
                              .collect(Collectors.toList());
System.out.println(lengths);  // Output: [4, 6]
```

---

107. Sum of Double Values Using Reduce

```
List<Double> doubles = List.of(1.1, 2.2, 3.3);
double sum = doubles.stream()
                    .reduce(0.0, Double::sum);
System.out.println(sum);  // Output: 6.6
```

---

108. Check if Any String is Empty

```
List<String> strings = List.of("hello", "", "world");
boolean anyEmpty = strings.stream()
                          .anyMatch(String::isEmpty);
System.out.println(anyEmpty);  // Output: true
```

---

109. Collect Even Numbers Into Set

```
List<Integer> numbers = List.of(1, 2, 3, 4, 4);
Set<Integer> evens = numbers.stream()
                            .filter(n -> n % 2 == 0)
                            .collect(Collectors.toSet());
System.out.println(evens);  // Output: [2, 4]
```

---

110. Concatenate Strings with Delimiter

```
List<String> words = List.of("red", "green", "blue");
String result = words.stream()
                     .collect(Collectors.joining(" | "));
System.out.println(result);  // Output: red | green | blue
```

Challenge Set 12 (111–120)

---

111. Find Minimum Number Using Reduce

```
List<Integer> numbers = List.of(10, 5, 3, 12);
int min = numbers.stream()
```

```
                .reduce(Integer.MAX_VALUE, Integer::min);
System.out.println(min);  // Output: 3
```

---

112. Filter Strings Containing a Substring

```
List<String> words = List.of("", "script", "python");
List<String> filtered = words.stream()
                             .filter(w -> w.contains(""))
                             .collect(Collectors.toList());
System.out.println(filtered);  // Output: [, script]
```

---

113. Convert List of Integers to Array

```
List<Integer> numbers = List.of(1, 2, 3);
Integer[] array = numbers.stream()
                         .toArray(Integer[]::new);
System.out.println(Arrays.toString(array));  // Output: [1, 2, 3]
```

---

114. Get Average of Int Stream

```
List<Integer> numbers = List.of(10, 20, 30);
OptionalDouble avg = numbers.stream()
                            .mapToInt(Integer::intValue)
                            .average();
avg.ifPresent(System.out::println);  // Output: 20.0
```

---

115. Check All Strings Have Length > 3

```
List<String> words = List.of("", "code", "stream");
boolean allLong = words.stream()
                       .allMatch(w -> w.length() > 3);
System.out.println(allLong);  // Output: true
```

---

116. Group Words by Length

```
List<String> words = List.of("apple", "bat", "cat", "dog", "elephant");
Map<Integer, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {3=[bat, cat, dog], 5=[apple], 8=[elephant]}
```

117. Generate Infinite Stream of Random Numbers (limit 5)

```
new Random().ints()
            .limit(5)
            .forEach(System.out::println);
```

---

118. Sort Strings by Last Character

```
List<String> words = List.of("apple", "banana", "carrot");
List<String> sorted = words.stream()
                           .sorted(Comparator.comparing(w ->
w.charAt(w.length() - 1)))
                           .collect(Collectors.toList());
System.out.println(sorted);  // Output: [banana, apple, carrot]
```

---

119. Count Frequency of Characters in a String

```
String input = "banana";
Map<Character, Long> freq = input.chars()
                                 .mapToObj(c -> (char)c)
                                 .collect(Collectors.groupingBy(c -> c,
Collectors.counting()));
System.out.println(freq);  // Output: {a=3, b=1, n=2}
```

---

120. Filter and Collect Distinct Integers

```
List<Integer> numbers = List.of(1, 2, 2, 3, 4, 4);
List<Integer> distinctEvens = numbers.stream()
                                     .filter(n -> n % 2 == 0)
                                     .distinct()
                                     .collect(Collectors.toList());
System.out.println(distinctEvens);  // Output: [2, 4]
```

Challenge Set 13 (121–130)

---

121. Find Max Length String

```
List<String> words = List.of("apple", "banana", "cherry");
String maxLen = words.stream()
                     .max(Comparator.comparingInt(String::length))
                     .orElse("");
System.out.println(maxLen);  // Output: banana
```

---

122. Convert List of Strings to Comma-Separated String

```
List<String> words = List.of("", "python", "c++");
String result = words.stream()
                     .collect(Collectors.joining(", "));
System.out.println(result);  // Output: , python, c++
```

---

123. Count Even Numbers in a List

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5, 6);
long count = numbers.stream()
                    .filter(n -> n % 2 == 0)
                    .count();
System.out.println(count);  // Output: 3
```

---

124. Find First String Starting with ‘A’

```
List<String> words = List.of("Bob", "Alice", "Adam");
Optional<String> firstA = words.stream()
                               .filter(w -> w.startsWith("A"))
                               .findFirst();
firstA.ifPresent(System.out::println);  // Output: Alice
```

---

125. Sum of Squares of Numbers

```
List<Integer> numbers = List.of(1, 2, 3, 4);
int sumSquares = numbers.stream()
                        .map(n -> n * n)
                        .reduce(0, Integer::sum);
System.out.println(sumSquares);  // Output: 30
```

---

126. Check If Any String Ends With ‘x’

```
List<String> words = List.of("box", "cat", "fox");
boolean anyEndsWithX = words.stream()
                            .anyMatch(w -> w.endsWith("x"));
```

```
System.out.println(anyEndsWithX);  // Output: true
```

---

127. Convert Stream to Set

```
List<String> words = List.of("apple", "banana", "apple");
Set<String> uniqueWords = words.stream()
                               .collect(Collectors.toSet());
System.out.println(uniqueWords);  // Output: [banana, apple]
```

---

128. Create Map of Word Lengths to Words

```
List<String> words = List.of("one", "two", "three", "four");
Map<Integer, List<String>> lengthMap = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(lengthMap);
// Output: {3=[one, two], 4=[four], 5=[three]}
```

---

129. Get Distinct Characters From a List of Strings

```
List<String> words = List.of("hello", "world");
List<Character> distinctChars = words.stream()
                                     .flatMap(w -> w.chars().mapToObj(c ->
(char)c))
                                     .distinct()
                                     .collect(Collectors.toList());
System.out.println(distinctChars);  // Output: [h, e, l, o, w, r, d]
```

---

130. Find the Longest Word Length

```
List<String> words = List.of("apple", "banana", "cherry");
int maxLength = words.stream()
                     .mapToInt(String::length)
                     .max()
                     .orElse(0);
System.out.println(maxLength);  // Output: 6
```

Challenge Set 14 (131–140)

---

131. Filter Strings Containing ‘a’ and Collect to List

```
List<String> words = List.of("apple", "banana", "cherry", "date");
List<String> filtered = words.stream()
                             .filter(w -> w.contains("a"))
                             .collect(Collectors.toList());
System.out.println(filtered);  // Output: [apple, banana, date]
```

---

132. Convert List of Integers to List of Strings

```
List<Integer> numbers = List.of(1, 2, 3);
List<String> strings = numbers.stream()
                              .map(String::valueOf)
                              .collect(Collectors.toList());
System.out.println(strings);  // Output: [1, 2, 3]
```

---

133. Find Minimum Value in List of Integers

```
List<Integer> numbers = List.of(10, 5, 8, 3);
int min = numbers.stream()
                 .min(Integer::compareTo)
                 .orElse(-1);
System.out.println(min);  // Output: 3
```

---

134. Sort List of Strings by Length Descending

```
List<String> words = List.of("apple", "banana", "cherry");
List<String> sorted = words.stream()

.sorted(Comparator.comparingInt(String::length).reversed())
                           .collect(Collectors.toList());
System.out.println(sorted);  // Output: [banana, cherry, apple]
```

---

135. Count Words with Length > 4

```
List<String> words = List.of("cat", "elephant", "dog", "giraffe");
long count = words.stream()
                  .filter(w -> w.length() > 4)
                  .count();
System.out.println(count);  // Output: 2
```

---

136. Check if All Numbers are Positive

```
List<Integer> numbers = List.of(1, 2, 3, 4);
boolean allPositive = numbers.stream()
                             .allMatch(n -> n > 0);
System.out.println(allPositive);  // Output: true
```

---

137. Convert List of Doubles to Sum

```
List<Double> doubles = List.of(1.1, 2.2, 3.3);
double sum = doubles.stream()
                    .mapToDouble(Double::doubleValue)
                    .sum();
System.out.println(sum);  // Output: 6.6
```

---

138. Group Strings by First Character

```
List<String> words = List.of("apple", "apricot", "banana", "blueberry");
Map<Character, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(w -> w.charAt(0)));
System.out.println(grouped);
// Output: {a=[apple, apricot], b=[banana, blueberry]}
```

---

139. Find Any String Containing ‘x’

```
List<String> words = List.of("box", "cat", "fox");
Optional<String> anyWithX = words.stream()
                                 .filter(w -> w.contains("x"))
                                 .findAny();
anyWithX.ifPresent(System.out::println);  // Output could be "box" or "fox"
```

---

140. Get Average of Integer List

```
List<Integer> numbers = List.of(10, 20, 30);
double avg = numbers.stream()
                    .mapToInt(Integer::intValue)
                    .average()
                    .orElse(0);
System.out.println(avg);  // Output: 20.0
```

Challenge Set 15 (141–150)

---

141. Convert List of Strings to Comma-Separated String

```
List<String> words = List.of("apple", "banana", "cherry");
String result = words.stream()
                     .collect(Collectors.joining(", "));
System.out.println(result);  // Output: apple, banana, cherry
```

---

142. Remove Duplicates from List of Integers

```
List<Integer> numbers = List.of(1, 2, 2, 3, 3, 3);
List<Integer> distinct = numbers.stream()
                                .distinct()
                                .collect(Collectors.toList());
System.out.println(distinct);  // Output: [1, 2, 3]
```

---

143. Convert List of Strings to Their Lengths and Sum Them

```
List<String> words = List.of("a", "bb", "ccc");
int totalLength = words.stream()
                       .mapToInt(String::length)
                       .sum();
System.out.println(totalLength);  // Output: 6
```

---

144. Filter Strings Ending With ‘e’

```
List<String> words = List.of("apple", "banana", "cake", "date");
List<String> endsWithE = words.stream()
                              .filter(w -> w.endsWith("e"))
                              .collect(Collectors.toList());
System.out.println(endsWithE);  // Output: [apple, cake, date]
```

---

145. Find Longest String

```
List<String> words = List.of("cat", "elephant", "dog");
String longest = words.stream()
                      .max(Comparator.comparingInt(String::length))
                      .orElse("");
System.out.println(longest);  // Output: elephant
```

---

146. Check If Any Number is Negative

```
List<Integer> numbers = List.of(1, 2, -3, 4);
boolean anyNegative = numbers.stream()
                             .anyMatch(n -> n < 0);
System.out.println(anyNegative);  // Output: true
```

---

147. Sort Numbers in Descending Order

```
List<Integer> numbers = List.of(5, 2, 9, 1);
List<Integer> sortedDesc = numbers.stream()
                                  .sorted(Comparator.reverseOrder())
                                  .collect(Collectors.toList());
System.out.println(sortedDesc);  // Output: [9, 5, 2, 1]
```

---

148. Group Strings by Their Length

```
List<String> words = List.of("cat", "dog", "bird", "elephant");
Map<Integer, List<String>> groupedByLength = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(groupedByLength);
// Output: {3=[cat, dog], 4=[bird], 8=[elephant]}
```

---

149. Count Strings That Contain ‘i’

```
List<String> words = List.of("this", "is", "a", "test");
long count = words.stream()
                  .filter(w -> w.contains("i"))
                  .count();
System.out.println(count);  // Output: 2
```

---

150. Find First Number Greater Than 10

```
List<Integer> numbers = List.of(3, 8, 15, 7);
Optional<Integer> firstGt10 = numbers.stream()
                                     .filter(n -> n > 10)
                                     .findFirst();
firstGt10.ifPresent(System.out::println);  // Output: 15
```

Challenge Set 16 (151–160)

---

151. Find Minimum Value in a List

```
List<Integer> numbers = List.of(5, 3, 8, 2);
int min = numbers.stream()
                 .min(Integer::compare)
                 .orElse(-1);
System.out.println(min);  // Output: 2
```

---

152. Check If All Strings Are Non-Empty

```
List<String> words = List.of("hello", "world", "");
boolean allNonEmpty = words.stream()
                           .allMatch(w -> !w.isEmpty());
System.out.println(allNonEmpty);  // Output: false
```

---

153. Create a Map of String to Its Length

```
List<String> words = List.of("", "stream", "lambda");
Map<String, Integer> map = words.stream()
                                .collect(Collectors.toMap(w -> w,
String::length));
System.out.println(map);
// Output: {=4, stream=6, lambda=6}
```

---

154. Count Frequency of Characters in a String

```
String input = "hello";
Map<Character, Long> freq = input.chars()
                                 .mapToObj(c -> (char) c)
                                 .collect(Collectors.groupingBy(c -> c,
Collectors.counting()));
System.out.println(freq);  // Output: {e=1, h=1, l=2, o=1}
```

---

155. Sum of Squares of Even Numbers

```
List<Integer> numbers = List.of(1, 2, 3, 4);
int sumSquares = numbers.stream()
                        .filter(n -> n % 2 == 0)
                        .mapToInt(n -> n * n)
                        .sum();
System.out.println(sumSquares);  // Output: 20 (4 + 16)
```

---

156. Convert Stream of Integers to Stream of Strings

```
List<Integer> numbers = List.of(1, 2, 3);
List<String> strings = numbers.stream()
                              .map(String::valueOf)
                              .collect(Collectors.toList());
System.out.println(strings);  // Output: [1, 2, 3]
```

---

157. Get List of Distinct Words from a Sentence

```
String sentence = "hello world hello ";
List<String> distinctWords = Arrays.stream(sentence.split(" "))
                                   .distinct()
                                   .collect(Collectors.toList());
System.out.println(distinctWords);  // Output: [hello, world, ]
```

---

158. Get Top 3 Largest Numbers

```
List<Integer> numbers = List.of(5, 8, 2, 9, 1);
List<Integer> top3 = numbers.stream()
                            .sorted(Comparator.reverseOrder())
                            .limit(3)
                            .collect(Collectors.toList());
System.out.println(top3);  // Output: [9, 8, 5]
```

---

159. Find Average of Double Values

```
List<Double> numbers = List.of(1.5, 2.5, 3.5);
double avg = numbers.stream()
                    .mapToDouble(Double::doubleValue)
                    .average()
                    .orElse(0);
System.out.println(avg);  // Output: 2.5
```

---

160. Filter and Collect Unique Strings That Start with "A"

```
List<String> words = List.of("Apple", "Ant", "Banana", "Apple");
List<String> uniqueA = words.stream()
                            .filter(w -> w.startsWith("A"))
                            .distinct()
                            .collect(Collectors.toList());
System.out.println(uniqueA);  // Output: [Apple, Ant]
```

Challenge Set 17 (161–170)

---

161. Convert List of Strings to Comma-Separated String

```
List<String> words = List.of("", "Streams", "Lambda");
String result = words.stream()
                     .collect(Collectors.joining(", "));
System.out.println(result);  // Output: , Streams, Lambda
```

---

162. Find Second Largest Number

```
List<Integer> numbers = List.of(3, 7, 2, 9, 5);
int secondLargest = numbers.stream()
                           .sorted(Comparator.reverseOrder())
                           .skip(1)
                           .findFirst()
                           .orElse(-1);
System.out.println(secondLargest);  // Output: 7
```

---

163. Check If Any String Contains "abc"

```
List<String> words = List.of("abcd", "efgh", "xyz");
boolean containsAbc = words.stream()
                           .anyMatch(w -> w.contains("abc"));
System.out.println(containsAbc);  // Output: true
```

---

164. Group Strings by Their Length

```
List<String> words = List.of("one", "two", "three", "four", "five");
Map<Integer, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {3=[one, two], 4=[four, five], 5=[three]}
```

---

165. Find Longest String Using Reduce

```
List<String> words = List.of("apple", "banana", "cherry");
String longest = words.stream()
```

```
                        .reduce((w1, w2) -> w1.length() > w2.length() ? w1 :
w2)
                        .orElse("");
System.out.println(longest);  // Output: banana
```

---

166. Collect List of Squares of Unique Numbers

```
List<Integer> numbers = List.of(1, 2, 2, 3);
List<Integer> squares = numbers.stream()
                              .distinct()
                              .map(n -> n * n)
                              .collect(Collectors.toList());
System.out.println(squares);  // Output: [1, 4, 9]
```

---

167. Partition Numbers into Even and Odd

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
Map<Boolean, List<Integer>> partitioned = numbers.stream()

.collect(Collectors.partitioningBy(n -> n % 2 == 0));
System.out.println(partitioned);
// Output: {false=[1, 3, 5], true=[2, 4]}
```

---

168. Find Sum of All Numbers Using Reduce

```
List<Integer> numbers = List.of(1, 2, 3, 4);
int sum = numbers.stream()
                 .reduce(0, Integer::sum);
System.out.println(sum);  // Output: 10
```

---

169. Filter Strings Ending With "ing"

```
List<String> words = List.of("running", "jump", "walking", "talk");
List<String> filtered = words.stream()
                             .filter(w -> w.endsWith("ing"))
                             .collect(Collectors.toList());
System.out.println(filtered);  // Output: [running, walking]
```

---

170. Convert List of Integers to Set

```
List<Integer> numbers = List.of(1, 2, 2, 3);
```

```
Set<Integer> numberSet = numbers.stream()
                                .collect(Collectors.toSet());
System.out.println(numberSet);  // Output: [1, 2, 3]
```

Challenge Set 18 (171–180)

---

171. Find Average of a List of Numbers

```
List<Integer> numbers = List.of(10, 20, 30, 40);
double average = numbers.stream()
                        .mapToInt(Integer::intValue)
                        .average()
                        .orElse(0);
System.out.println(average);  // Output: 25.0
```

---

172. Find Maximum Value in List Using Comparator

```
List<Integer> numbers = List.of(5, 9, 2, 15, 3);
int max = numbers.stream()
                 .max(Comparator.naturalOrder())
                 .orElse(-1);
System.out.println(max);  // Output: 15
```

---

173. Count Strings That Start With a Specific Letter

```
List<String> names = List.of("Alice", "Bob", "Anna", "Charlie");
long count = names.stream()
                  .filter(n -> n.startsWith("A"))
                  .count();
System.out.println(count);  // Output: 2
```

---

174. Convert Stream to Array

```
List<String> fruits = List.of("Apple", "Banana", "Cherry");
String[] array = fruits.stream()
                       .toArray(String[]::new);
System.out.println(Arrays.toString(array));  // Output: [Apple, Banana,
Cherry]
```

---

175. Sort Strings by Last Character

```
List<String> words = List.of("apple", "banana", "carrot");
List<String> sorted = words.stream()
                           .sorted(Comparator.comparingInt(w ->
w.charAt(w.length() - 1)))
                           .collect(Collectors.toList());
System.out.println(sorted);  // Output: [banana, apple, carrot]
```

---

176. Find All Distinct Characters From a List of Words

```
List<String> words = List.of("apple", "banana");
List<Character> distinctChars = words.stream()
                                     .flatMap(w -> w.chars().mapToObj(c ->
(char) c))
                                     .distinct()
                                     .collect(Collectors.toList());
System.out.println(distinctChars);
// Output: [a, p, l, e, b, n]
```

---

177. Join List of Integers as String with Dash Separator

```
List<Integer> numbers = List.of(1, 2, 3);
String joined = numbers.stream()
                       .map(String::valueOf)
                       .collect(Collectors.joining("-"));
System.out.println(joined);  // Output: 1-2-3
```

---

178. Find Number of Empty Strings

```
List<String> strings = List.of("abc", "", "def", "", "");
long emptyCount = strings.stream()
                         .filter(String::isEmpty)
                         .count();
System.out.println(emptyCount);  // Output: 3
```

---

179. Check if All Strings Have Length > 3

```
List<String> words = List.of("", "stream", "lambda");
boolean allLongerThan3 = words.stream()
                              .allMatch(w -> w.length() > 3);
System.out.println(allLongerThan3);  // Output: true
```

---

180. Find First String Containing "test"

```
List<String> texts = List.of("foo", "bar", "testing", "baz");
String found = texts.stream()
                    .filter(s -> s.contains("test"))
                    .findFirst()
                    .orElse("Not found");
System.out.println(found);  // Output: testing
```

Challenge Set 19 (181–190)

---

181. Convert List of Integers to Their Squares and Collect in a Set

```
List<Integer> numbers = List.of(1, 2, 3, 2);
Set<Integer> squares = numbers.stream()
                              .map(n -> n * n)
                              .collect(Collectors.toSet());
System.out.println(squares);  // Output: [1, 4, 9]
```

---

182. Find the Longest String Using reduce()

```
List<String> words = List.of("cat", "elephant", "dog");
String longest = words.stream()
                      .reduce((w1, w2) -> w1.length() > w2.length() ? w1 :
w2)
                      .orElse("");
System.out.println(longest);  // Output: elephant
```

---

183. Partition Numbers into Even and Odd Using partitioningBy

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
Map<Boolean, List<Integer>> partitioned = numbers.stream()

.collect(Collectors.partitioningBy(n -> n % 2 == 0));
System.out.println(partitioned);
// Output: {false=[1, 3, 5], true=[2, 4]}
```

---

184. Use peek() to Debug Stream Processing

```
List<String> fruits = List.of("apple", "banana", "cherry");
List<String> result = fruits.stream()
```

```
                                .peek(f -> System.out.println("Original: " +
f))
                                .map(String::toUpperCase)
                                .peek(f -> System.out.println("Uppercase: " +
f))
                                .collect(Collectors.toList());
System.out.println(result);
// Output: Original and Uppercase printed for each fruit, then [APPLE,
BANANA, CHERRY]
```

---

185. Sum of All Odd Numbers

```
List<Integer> numbers = List.of(1, 2, 3, 4, 5);
int sumOdd = numbers.stream()
                    .filter(n -> n % 2 != 0)
                    .mapToInt(Integer::intValue)
                    .sum();
System.out.println(sumOdd);  // Output: 9
```

---

186. Group Strings by Length

```
List<String> words = List.of("cat", "dog", "elephant", "rat");
Map<Integer, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(String::length));
System.out.println(grouped);
// Output: {3=[cat, dog, rat], 8=[elephant]}
```

---

187. Find First Even Number or Default to -1

```
List<Integer> numbers = List.of(1, 3, 5, 6);
int firstEven = numbers.stream()
                       .filter(n -> n % 2 == 0)
                       .findFirst()
                       .orElse(-1);
System.out.println(firstEven);  // Output: 6
```

---

188. Create a Map from List of Strings with String Length as Value

```
List<String> words = List.of("apple", "banana", "cherry");
Map<String, Integer> map = words.stream()
                                .collect(Collectors.toMap(w -> w,
String::length));
System.out.println(map);
// Output: {apple=5, banana=6, cherry=6}
```

189. Count Frequency of Each Character in a String

```
String input = "hello world";
Map<Character, Long> freq = input.chars()
                                 .mapToObj(c -> (char) c)
                                 .filter(c -> c != ' ')
                                 .collect(Collectors.groupingBy(c -> c,
Collectors.counting()));
System.out.println(freq);
// Output: {d=1, e=1, h=1, l=3, o=2, r=1, w=1}
```

---

190. Collect Top 3 Longest Strings

```
List<String> words = List.of("apple", "banana", "cherry", "date");
List<String> top3 = words.stream()

.sorted(Comparator.comparingInt(String::length).reversed())
                        .limit(3)
                        .collect(Collectors.toList());
System.out.println(top3);  // Output: [banana, cherry, apple]
```

Challenge Set 20 (191–200)

---

191. Convert List of Strings to Comma-Separated String

```
List<String> words = List.of("apple", "banana", "cherry");
String result = words.stream()
                     .collect(Collectors.joining(", "));
System.out.println(result);  // Output: apple, banana, cherry
```

---

192. Find Minimum Integer in a List Using reduce()

```
List<Integer> numbers = List.of(5, 3, 9, 1, 4);
int min = numbers.stream()
                 .reduce(Integer::min)
                 .orElse(-1);
System.out.println(min);  // Output: 1
```

---

193. Create a Map of Strings to Their First Character

```
List<String> words = List.of("apple", "banana", "cherry");
Map<String, Character> map = words.stream()
                                  .collect(Collectors.toMap(w -> w, w ->
w.charAt(0)));
System.out.println(map);
// Output: {apple=a, banana=b, cherry=c}
```

---

194. Filter Out Null Values from List

```
List<String> list = Arrays.asList("apple", null, "banana", null, "cherry");
List<String> filtered = list.stream()
                            .filter(Objects::nonNull)
                            .collect(Collectors.toList());
System.out.println(filtered);  // Output: [apple, banana, cherry]
```

---

195. Find Average Length of Strings

```
List<String> words = List.of("apple", "banana", "cherry");
double avgLength = words.stream()
                        .mapToInt(String::length)
                        .average()
                        .orElse(0);
System.out.println(avgLength);  // Output: 6.0
```

---

196. Check if Any String Contains “a”

```
List<String> words = List.of("apple", "banana", "cherry");
boolean anyContainA = words.stream()
                           .anyMatch(w -> w.contains("a"));
System.out.println(anyContainA);  // Output: true
```

---

197. Count How Many Strings Start With “b”

```
List<String> words = List.of("apple", "banana", "blueberry", "cherry");
long countB = words.stream()
                   .filter(w -> w.startsWith("b"))
                   .count();
System.out.println(countB);  // Output: 2
```

---

198. Create a List of Lengths for Each String

```
List<String> words = List.of("apple", "banana", "cherry");
List<Integer> lengths = words.stream()
                             .map(String::length)
                             .collect(Collectors.toList());
System.out.println(lengths);  // Output: [5, 6, 6]
```

---

199. Sort Strings Alphabetically Ignoring Case

```
List<String> words = List.of("banana", "Apple", "cherry");
List<String> sorted = words.stream()
                           .sorted(String.CASE_INSENSITIVE_ORDER)
                           .collect(Collectors.toList());
System.out.println(sorted);  // Output: [Apple, banana, cherry]
```

---

200. Create a Map Grouping Strings by Their Last Character

```
List<String> words = List.of("apple", "banana", "cherry", "date");
Map<Character, List<String>> grouped = words.stream()

.collect(Collectors.groupingBy(w -> w.charAt(w.length() - 1)));
System.out.println(grouped);
// Output: {e=[apple, date], a=[banana], y=[cherry]}
```
