<table>
<tr><th>Function</th><th>Time Complexity</th></tr>
<tr><td>public static <E> boolean unique(List<E> givenList) {
for (int currentItem = 0; currentItem < givenList.size(); currentItem++) // Loop
for (int nextItem = currentItem + 1; nextItem < givenList.size(); nextItem++)
if (givenList.get(currentItem) == givenList.get(nextItem))
return false;

return true;
}</td><td>$O(n^2)$</td></tr>
<tr><td>public static List<Integer> allMultiples(List<Integer> givenList, int divisor) {
List<Integer> outputList = new ArrayList<>();

for (int curNumber : givenList) // Loop every number in the given list
if (curNumber % divisor == 0) // Check if curNumber is divislbe by divis
outputList.add(curNumber);

return outputList;
}</td><td>$O(n)$</td></tr>
<tr><td>public static List<String> allStringsOfSize(List<String> givenList, int length) {
List<String> outputList = new ArrayList<>();

for (String curString : givenList) // Loop through every string in the list.
if (curString.length() == length) // If current string's length is the sa
outputList.add(curString);

return outputList;
}</td><td>$O(n)$</td></tr>
<tr><td>public static <E> boolean isPermutation(List<E> givenListOne, List<E> givenListTwo) {
if (givenListOne.size() != givenListTwo.size() || givenListOne.isEmpty()) // Chec
return false;

List<E> listTwoCopy = new ArrayList<>(givenListTwo); // Copy the second list so w

for (E curObject : givenListOne) // Loop through every object in the first list.
if (!listTwoCopy.remove(curObject)) // Try to remove the first list's object
return false; // If remove returns false, it didn't have it.

return true; // All of our checks have passed. They are permutations.
}</td><td>$O(n^2)$</td></tr>
<tr><td>public static List<String> stringToListOfWords(String givenString) {
List<String> outputList = new ArrayList<>();

for (String curString : givenString.split(regex: "\\s+")) // Loop
outputList.add(curString.replaceAll(regex: "\\W", replacement:

return outputList;
}</td><td>$O(n)$</td></tr>
<tr><td>public static <E> void removeAllInstances(List<E> givenList, E givenItem) {
while (givenList.contains(givenItem)) // Keep looping until the list no
givenList.remove(givenItem); // Removal of object.
}</td><td>$O(n^2)$</td></tr>
</table>