1. One possible solution appears below.

```java
public static void printJumbles(Dictionary words, List<String> options, int numWords) {
    printJumbles(words, options, numWords, "");
}

public static void printJumbles(Dictionary words, List<String> options, int choices, String soFar) {
    if (choices == 0) {
        if (words.contains(soFar)) {
            System.out.println(soFar);
        }
    } else {
        for (String word : options) {
            printJumbles(words, options, choices - 1, soFar + word);
        }
    }
}
```

2. One possible solution appears below.

```java
public static void printJumbles2(Dictionary words, List<String> options) {
    printJumbles2(words, options, "");
}

// pre : word
public static void printJumbles2(Dictionary words, List<String> options, String soFar) {
    if (words.contains(soFar)) {
        System.out.println(soFar);
    }
    for (String word : options) {
        String newString = soFar + word;
        if (words.containsPrefix(newString)) {
            printJumbles2(words, options, newString);
        }
    }
}
```

3. One possible solution appears below.

```java
public static void printJumbles3(Dictionary words, List<String> options) {
    printJumbles3(words, options, "", new TreeSet<>());
}

public static void printJumbles3(Dictionary words, List<String> options, String soFar, Set<String> used) {
    if (words.contains(soFar)) {
        System.out.println(soFar);
    }
    for (String word : options) {
        String newString = soFar + word;
        if (!used.contains(word) && words.containsPrefix(newString)) {
            used.add(word);
            printJumbles3(words, options, newString, used);
            used.remove(word);
        }
    }
```
In terms of efficiency, when working with the 50 state abbreviations, the
printJumbles method calls the dictionary's contains method 125,000 times. The
printJumbles2 method calls the contains method 790 times and the containsPrefix
method 39,500 times.

For students who are interested, the complete code for Dictionary.java and the
sample solution Jumble.java are available along with the data files
dictionary.txt and states.txt.

https://courses.cs.washington.edu/courses/cse143x/23au/lectures/Jumble.java
https://courses.cs.washington.edu/courses/cse143x/23au/lectures/dictionary.txt
https://courses.cs.washington.edu/courses/cse143x/23au/lectures/states.txt