Generate

Rar the Cat has a word **W** containing **N** distinct lowercase alphabets ('a' to 'z').

Rar the Cat would like you to generate all possible permutations of the **N** alphabets followed by all possible sub-sequences of the **N** alphabets.

A permutation of the alphabets in **W** is generated by reordering the alphabets in **W**. You are to print all possible such re-orderings. For example, permutations of 'cat' are 'cta', 'cat', 'atc', 'act', 'tac', 'tca'.

A sub-sequence of the alphabets in **W** is generated by choosing a non-empty subset of alphabets in **W**, without changing their relative order in **W**. For example, sub-sequences of 'cat' are 'c', 'a', 't', 'ca', 'at', 'ct', 'cat'.

The permutations should be printed in increasing lexicographical order, followed by the sub-sequences in increasing lexicographical order as well. In order to sort a list of strings in lexicographical order, you can simply utilize **Collections.sort** on a list of **String**s. The default *compareTo* function in String class is already comparing in lexicographical order.

Input
The input contains a single line, containing the word **W** of length **N**.

Output
The output should contain **(N!) + 2^N – 1** lines.
The first **N!** lines should contain all possible permutations of the alphabets in W, printed in increasing lexicographical order.
The next **2^N-1** lines should contain all possible sub-sequences of the alphabets in W, printed in increasing lexicographical order as well.

Limits
- $1 \leq \mathbf{N} \leq 9$
- **W** will only contain distinct lowercase alphabets ('a' to 'z').

Sample Input (**generate1.in**)	Sample Output (**generate1.out**)

`tan`	`ant`
	`atn`
	`nat`
	`nta`
	`tan`
	`tna`
	`a`
	`an`
	`n`
	`t`
	`ta`
	`tan`
	`tn`

Sample Input (**generate2.in**)	Sample Output (**generate2.out**)
`f`	`f` `f`

Notes:

1. You should develop your program in the subdirectory **ex1** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – Generate.java

You are given the below skeleton file `Generate.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
 * Name         :
 * Matric. No   :
 * PLab Acct.   :
 */

import java.util.*;

public class Generate {
    private void run() {
        //implement your "main" method here
    }

    public static void main(String[] args) {
        Generate newGenerate = new Generate();
        newGenerate.run();
    }
}
```