CPE 349 – Permutations

Deliverable:

Source code for a single class, **CombObjects.java** with the methods described below. This will be submitted on unix using the handin command.

Generating combinatorial objects is fundamental to many problems in computer science. In this assignment you will implement algorithms to: 1. generate Gray codes as describe in lab and 2. generate the permutations of a set in lexicographic order. 3. Generate permutations in an order where the adjacent permutation differ only in the exchange of two adjacent entries. (Why? E.g. This enables different paths in TSP to be computed by only a few changes in the previous path length rather than computing each path length from scratch.)

Your Class, **CombObjects.java**, must meet the following specifications:

- Implement a method **getGrayCode(int n)** that returns an ArrayList of strings where each string is a bitstring (contains only the characters 0 and 1) that represent the subsets of a set containing n elements. Your method must return the Gray Code as described in the previous lab.
- Implement a method **getLexPerm (String str)** that returns an ArrayList of Strings in lexicographic order. The input argument can be assumed to be a string of distinct lower case letters (in alphabetical order). You may assume the input is correct - represents a set of distinct letters in order, e.g. *abcd*.
- Implement a method **getMinChgPerm (String str)** that returns an ArrayList of Strings that satisfy a minimum change requirement. Again the input argument can be assumed to be a string of distinct lower case letters (in alphabetical order). You may assume the input is correct - represents a set of distinct letters in order, e.g. *abcd*.
- Your program must be well structured, commented, and easy to read.
- All three methods must be **recursive** and the **<u>last two must follow the high level description below or they may not pass the tests</u>**. See the attached for the desired output for "abc" and "abcd"

High level description of recursive algorithm to generate permutations in lexicographic order

```
//  Assumes string contains characters in appropriate order
If the string is empty return it

Loop through all character positions of the string containing the characters to be
permuted, for each character
      Form a simpler word by removing the character
      Generate all permutations of the simpler word recursively
      Add the removed character to the front of each permutation of the simpler word, and
            add the resulting permutation to a list
Return all these newly constructed permutations
```

High level description of recursive algorithm to generate permutations satisfying the minimum change requirement

```
//  Assumes string contains characters in appropriate order
If the string is empty return it
Remove the last character, call it x, of the string
Generate all permutations (satisfying min change requirement) of the simpler word
Loop over the returned permutations
   • insert the removed character into a returned permutation into all possible
     positions moving right to left
   • insert the removed character into the next returned permutation into all possible
     positions moving left to right
Return all these newly constructed permutations
```

Thus in the example below for abc -

the method would remove the c, call itself on ab
the call would return an array list contain ab and ba
then the loop would execute only once.
the first line (a loop) would create abc, acb, cab by inserting the c right to left into ab
the next line (also a loop) would create cba, bca, bac by inserting the c left to right into ba

Example input for getLexPerm: abc

Output:
```
abc
acb
bac
bca
cab
cba
```

Example input for getMinChgPerm: abc

Output:
```
abc
acb
cab
cba
bca
bac
```

Example input for getLexPerm: abcd

Output:
```
abcd
abdc
acbd
acdb
adbc
adcb
bacd
badc
bcad
bcda
bdac
bdca
cabd
cadb
cbad
cbda
cdab
cdba
dabc
dacb
dbac
dbca
dcab
dcba
```

Example input for getMinChgPerm: abcd

Output:
```
abcd
abdc
adbc
dabc
dacb
adcb
acdb
acbd
cabd
cadb
cdab
dcab
dcba
cdba
cbda
cbad
bcad
bcda
bdca
dbca
dbac
bdac
badc
bacd
```

Please submit **CombObjects.java** Monday Oct. 12 by 9 pm.

By using "handin gradertk cpe349assign2 CombObjects.java"