Given a string of length n, print all permutation of the given string. Repetition of characters is allowed. Print these permutations in lexicographically sorted order
Examples:

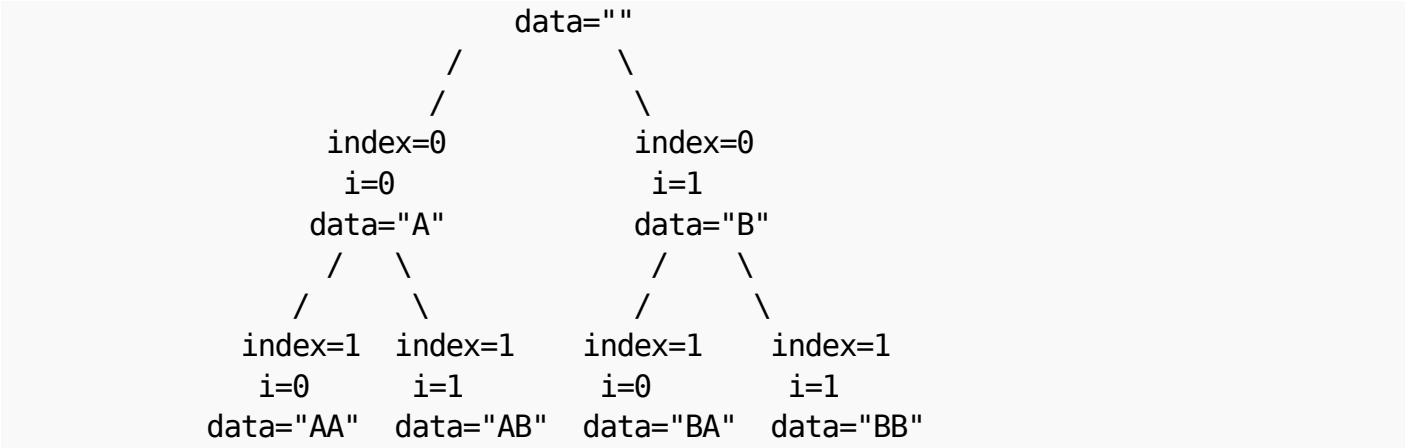
```
Input: AB
Ouput: All permutations of AB with repetition are:
      AA
      AB
      BA
      BB

Input: ABC
Output: All permutations of ABC with repetition are:
       AAA
       AAB
       AAC
       ABA
       ...
       ...
       CCB
       CCC
```

For an input string of size n, there will be n^n permutations with repetition allowed. The idea is to fix the first character at first index and recursively call for other subsequent indexes. Once all permutations starting with the first character are printed, fix the second character at first index. Continue these steps till last character. Thanks to PsychoCoder for providing following C implementation.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

/* Following function is used by the library qsort() function to sort an
  array of chars */
int compare (const void * a, const void * b);

/* The main function that recursively prints all repeated permutations of
  the given string. It uses data[] to store all permutations one by one */
void allLexicographicRecur (char *str, char* data, int last, int index)
{
    int i, len = strlen(str);

    // One by one fix all characters at the given index and recur for the
    // subsequent indexes
    for ( i=0; i<len; i++ )
    {
        // Fix the ith character at index and if this is not the last index
        // then recursively call for higher indexes
        data[index] = str[i] ;

        // If this is the last index then print the string stored in data[]
        if (index == last)
            printf("%s\n", data);
        else // Recur for higher indexes
            allLexicographicRecur (str, data, last, index+1);
    }
}

/* This function sorts input string, allocate memory for data (needed for
  allLexicographicRecur()) and calls allLexicographicRecur() for printing all
  permutations */
```

```
void allLexicographic(char *str)
{
    int len = strlen (str) ;

    // Create a temp array that will be used by allLexicographicRecur()
    char *data = (char *) malloc (sizeof(char) * (len + 1)) ;
    data[len] = '\0';

    // Sort the input string so that we get all output strings in
    // lexicographically sorted order
    qsort(str, len, sizeof(char), compare);

    // Now print all permutaions
    allLexicographicRecur (str, data, len-1, 0);

    // Free data to avoid memory leak
    free(data);
}

// Needed for library function qsort()
int compare (const void * a, const void * b)
{
    return ( *(char *)a - *(char *)b );
}

// Driver program to test above functions
int main()
{
    char str[] = "ABC";
    printf("All permutations with repetition of %s are: \n", str);
    allLexicographic(str);
    getchar();
    return 0;
}
```

Following is recursion tree for input string “AB”. The purpose of recursion tree is to help in understanding the above implementation as it shows values of different variables.



In the above implementation, it is assumed that all characters of the input string are different. The implementation can be easily modified to handle the repeated characters. We have to add a preprocessing step to find unique characters (before calling allLexicographicRecur()).