Explanation of permutations of a string

Given a string, print all permutations of it.

> **Input:** str = "ABC"
>
> **Output** : ABC ACB BAC BCA CAB CBA

> **Idea:** We iterate from first to last index. For every index i, we swap the i-th character with the first index. This is how we fix characters at the current first index, then we recursively generate all permutations beginning with fixed characters (by parent recursive calls). After we have recursively generated all permutations with the first character fixed, then we move the first character back to its original position so that we can get the original string back and fix the next character at first position.

Illustration:

We swap 'A' with 'A'. Then we recursively generate all permutations beginning with A. While returning from the recursive calls, we revert the changes made by them using the same swap again. So we get the original string "ABC".

Then we swap 'A' with 'B' and generate all permutations beginning with 'B'.

Similarly, we generate all permutations beginning with 'C'
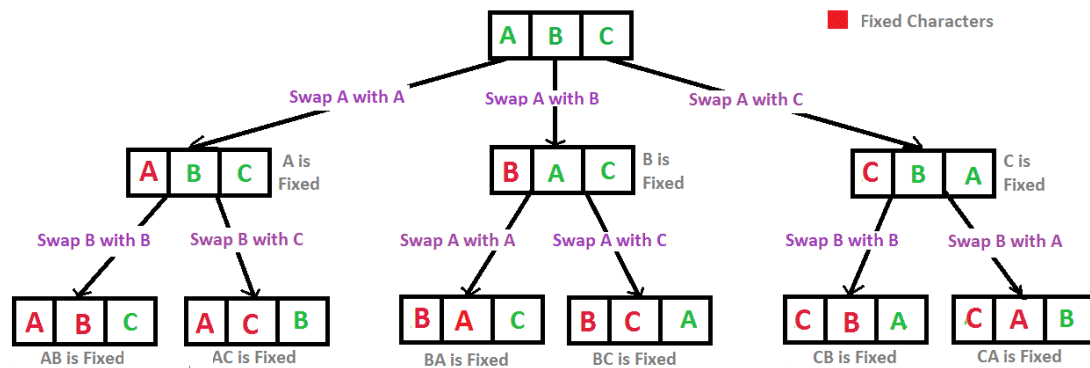
Recursion Tree for Permutations of String "ABC"

```cpp
// C++ program to print all
// permutations with duplicates allowed
#include <bits/stdc++.h>
using namespace std;

/* Function to print permutations of string
This function takes three parameters:
1. String
2. Starting index of the string
3. Ending index of the string. */
void permute(string &str, int l, int r)
{
    if (l == r)
        cout << str << " ";
    else    {
        for (int i = l; i <= r; i++)
        {
            swap(str[l], str[i]);
            permute(str, l+1, r);
            swap(str[l], str[i]);
        }
    }
}

/* Driver program to test above functions */
int main()
{
```

```
    string str = "ABC";
    permute(str, 0, str.length()-1);
    return 0;
}
```

Output:

ABC ACB BAC BCA CBA CAB

How do we get permutations in lexicographically sorted order?

We can put all permutations in an array instead of printing them immediately. Then we can sort the array and print the array.