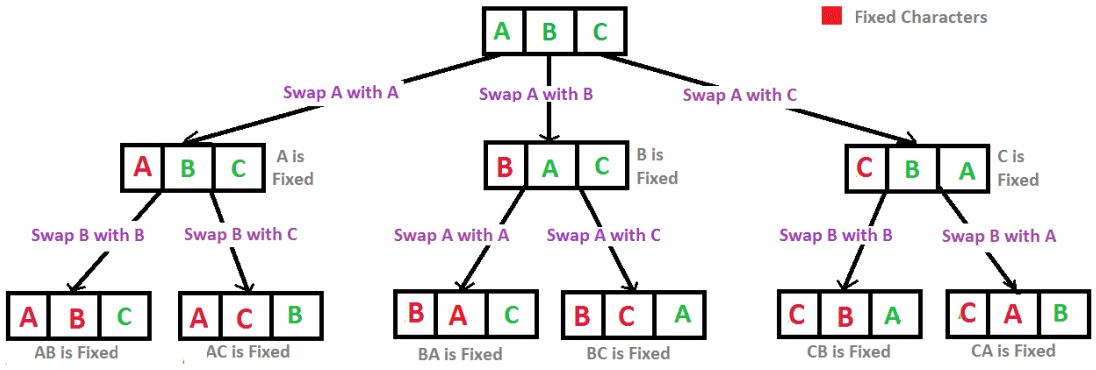
Write a program to print all permutations of a given string

A permutation, also called an “arrangement number” or “order,” is a rearrangement of the elements of an ordered list S into a one-to-one correspondence with S itself. A string of length n has n! permutation.
Source: Mathword(http://mathworld.wolfram.com/Permutation.html)

Below are the permutations of string ABC.
ABC ACB BAC BCA CBA CAB

Recommended: Please solve it on “*PRACTICE* ” first, before moving on to the solution.

Here is a solution that is used as a basis in backtracking.



Recursion Tree for Permutations of String "ABC"

```
// C program to print all permutations with duplicates allowed
#include <stdio.h>
#include <string.h>

/* Function to swap values at two pointers */
void swap(char *x, char *y)
{
```

```
    char temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

/* 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(char *a, int l, int r)
{
   int i;
   if (l == r)
     printf("%s\n", a);
   else
   {
       for (i = l; i <= r; i++)
       {
          swap((a+l), (a+i));
          permute(a, l+1, r);
          swap((a+l), (a+i)); //backtrack
       }
   }
}

/* Driver program to test above functions */
int main()
{
    char str[] = "ABC";
    int n = strlen(str);
    permute(str, 0, n-1);
    return 0;
}
```

Run on IDE

Java

```
// Java program to print all permutations of a
// given string.
public class Permutation
{
    public static void main(String[] args)
    {
        String str = "ABC";
        int n = str.length();
        Permutation permutation = new Permutation();
        permutation.permute(str, 0, n-1);
    }

    /**
     * permutation function
     * @param str string to calculate permutation for
     * @param l starting index
     * @param r end index
     */
    private void permute(String str, int l, int r)
    {
        if (l == r)
            System.out.println(str);
        else
        {
            for (int i = l; i <= r; i++)
            {
                str = swap(str,l,i);
```

```
                permute(str, l+1, r);
                str = swap(str,l,i);
            }
        }
    }

    /**
     * Swap Characters at position
     * @param a string value
     * @param i position 1
     * @param j position 2
     * @return swapped string
     */
    public String swap(String a, int i, int j)
    {
        char temp;
        char[] charArray = a.toCharArray();
        temp = charArray[i] ;
        charArray[i] = charArray[j];
        charArray[j] = temp;
        return String.valueOf(charArray);
    }

}

// This code is contributed by Mihir Joshi
```

Run on IDE

Python

```
# Python program to print all permutations with
# duplicates allowed

def toString(List):
    return ''.join(List)

# 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.
def permute(a, l, r):
    if l==r:
        print toString(a)
    else:
        for i in xrange(l,r+1):
            a[l], a[i] = a[i], a[l]
            permute(a, l+1, r)
            a[l], a[i] = a[i], a[l] # backtrack

# Driver program to test the above function
string = "ABC"
n = len(string)
a = list(string)
permute(a, 0, n-1)

# This code is contributed by Bhavya Jain
```

Run on IDE

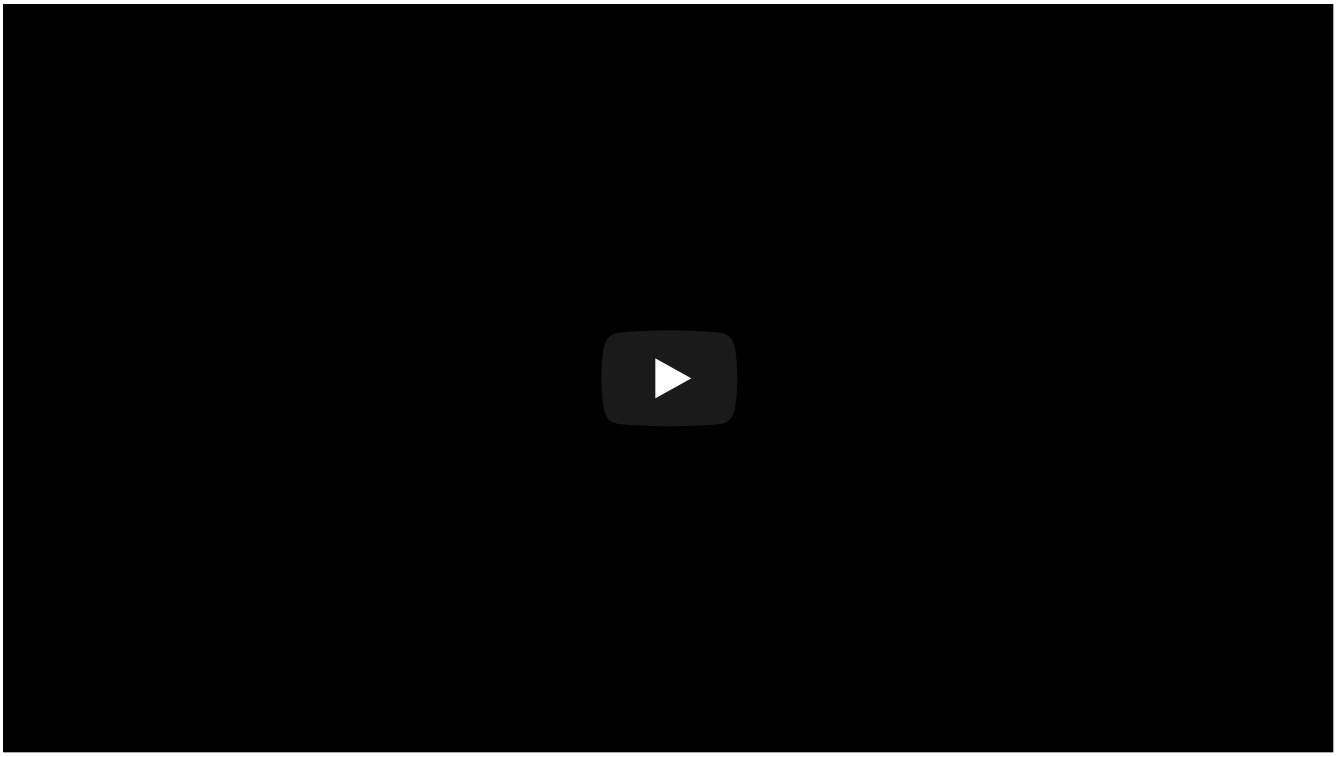
Output:

```
ABC
ACB
BAC
BCA
CBA
CAB
```

Algorithm Paradigm: Backtracking
Time Complexity: O(n*n!) Note that there are n! permutations and it requires O(n) time to print a a permutation.

Note : The above solution prints duplicate permutations if there are repeating characters in input string. Please see below link for a solution that prints only distinct permutations even if there are duplicates in input.

Print all distinct permutations of a given string with duplicates.
Permutations of a given string using STL



Asked in: **Accolite**,**Amazon**,**Cisco**,**MAQ-Software**,**Samsung**

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.