Date : Page No :

Assignment Number

Problem Statement

Program in C to print all possible permutations of the characters of a string.

Theory

The notion of **permutation** relates to the act of **arranging** all the members of a set into some sequence or order, or if the set is already ordered, **rearranging** (reordering) its elements, a process called **permuting**. These differ from combinations, which are selections of some members of a set where order is disregarded. For example, written as tuples, there are six permutations of the set {1,2,3}, namely: (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), and (3,2,1). These are all the possible orderings of this three element set. Hence, for a given string of finite length, permuting it would be rearranging all of its characters in every combination possible, irrespective of their order. The general mathematical formula for number of permutations of k elements from a set of n elements is the following :

$$^{n}P_{r} = \frac{n!}{(n-k)!}$$

Since we are taking all characters of the string of length n, for our case, n = k. Hence the denomiator (n – k)! will be 0! = 1. Hence, number of permutations will be n! .

Example : Consider the given string “nil”, without the quotes. All of its possible permutations of the character set {n, i, l} are : {n i l, n l i, i n l, i l n, l i n, l n i}.

Algorithms

Algorithm for PermuteString(s, i, m)

Input :

1. The string to permute, say S.
2. The index to start permutation from, say i
3. The index to end permutation at, say m

Output : All possible permutations of the given string between ith and mth index.

Steps :

Step 1: If(i == m)

Then

A) Set j = 1

B) Repeat through step B.i to B.ii while(j <= m)

i. Print " ", s[j]

ii. Set j = j+1

[End of while loop]

C) Print "\n”

Step 2: Else

A) Set j = i

B) Repeat through step B.i to B.iv while(j <= m)

i. Set c = s[i], s[i] = s[j], s[j] = c

ii. Call PermuteString(s, i + 1, m)

iii. Set c = s[i], s[i] = s[j], s[j] = c

iv. Set j = j + 1

[End of while loop]

[End of if structure]

Algorithm or Main()

1. Print "Enter the string : "
2. Input str
3. Print "The required permutations are : "
4. Call PermuteString(str, 1, StringLength(str)) // StringLength is a function
   // which calculates the length
   // of the given string

Source Code

```
#include <stdio.h>
#include <string.h>
/* Function to show the permutations of the given string */
void permute(char *s, int i, int m)  {
    int j;
    char c;
    if (i == m) {
        for (j = 0; j < m; j++) // Printing each permutation with every call
            printf(" %c", s[j]);
        printf("\n");
    } else {
        for (j = i; j < m; j++) {
            // Swapping a[i] and a[j]
            c = s[i];
            s[i] = s[j];
            s[j] = c;
            permute(s, i + 1, m); // Recursive call
            // Swapping back
            c = s[i];
            s[i] = s[j];
```

```
            s[j] = c;
        }
    }
}
/* Driver */
int main() {
    char str[100];
    printf("\nEnter the string : ");
    scanf("%s", str);
    printf("\nThe required permutations are : \n");
    permute(str, 0, strlen(str)); // Calling the permute function
    return 0;
}
```

Input and Output

Set 1 :

Enter the string : nil

The required permutations are :

n i l

n l i

i n l

i l n

l i n

l n i

Set 2 :

Enter the string :

The required permutations are :

c o o l

c o l o

c o o l

c o l o

c l o o

c l o o

o c o l

o c l o

o o c l

o o l c

o l o c

o l c o

o o c l

o o l c

o c o l

o c l o

o l c o

o l o c

l o o c

l o c o

l o o c

l o c o

l c o o

l c o o

Discussion

1. For large strings, number of output permutations will be countably infinite.
2. Since the program is recursively implemented, there might be a stack overflow if a string of large size is given.