Question :

Task -1:
Propose an algorithm that will print all the permutations of a given string. Explain the working methodology of your algorithm in detail.
Sample Input 1: abc
Sample output 1:
abc
acb
bac
bca
cab
cba
Sample Input 2:
abb
Sample output 2:
abb
bab
bba

Task 2:
What modifications do you need to do in the above-proposed algorithm to handle combinations if the number of selected items k is given.

Sample Input 3: abc
and k = 2
Sample output 3:
ab
ac
bc

Submission guideline:
You need to submit a pdf file.
Word limit: 500 (excluding figures, tables, etc.)

Answer :

Task 1 :

```
#include <bits/stdc++.h>

using namespace std;

bool d_swap(char str[], int s, int c)

{

   for (int i = s; i < c; i++)

   {

           if (str[i] == str[c]){

                   return 0;

           }

    }

   return 1;

}

void perm(char str[], int left, int right)

{

   if (left >= right) {

      cout << "\t" << str << endl;

      return;

   }


   for (int i = left; i < right; i++) {

     bool check = d_swap(str, left, i);

      if (check) {

         swap(str[left], str[i]);

         perm(str, left + 1, right);

         swap(str[idx], str[i]);

      }

   }
```

```
}
int main()
{
   char str[1000] ;
   cout << "Enter the string : " << endl ;
   scanf("%s",str);
   cout << "All possible permutaions : " << endl ;
   int n = strlen(str);
   perm(str, 0, n);

   return 0;
}
```

Explanation :

At first complier will take a string as an input and will store the size of string in **n** . Then it will call the function **perm** (and pass the parameters) where it is going to execute the permutation of the string . If left index == right index , then it will print the permutation . Else , to check all the permutation it will create a for loop where (i=left index ; i<right index ; i++) this code will swap the each indexes and will do recursion to backtrack to see another way and again it will do swapping for the last time to reach the leaf node . To avoid duplication , I created a function called **d_swap** where it will at first check if any of the characters are repeated or not and then it will do our permutation .**Example :** Let's take a input **abb.** Here , **length of the string = 3** . So , main function will pass the string , **left index=0** and **right index=3** to the **perm** function . At first it will check if **left index == right index** or not . If not then it will check if there is any kind of duplication or not . The function **d_swap** will return **true** cause i=0 and left index=0 and there is not duplication . Then it will go into the if statement and the indexes will get swapped and for backtrack we will use recursion and it will get swapped again to reach the leaf node and will get out of the for loop and will show output .If it finds any duplication in the string the **d_swap** function will show **false** and there won't happen any swapping but it will **increase the value of i** .This process will **continue until i<3 .**

Output :

abb

bab

bba

Task 2 :

```
#include <bits/stdc++.h>
using namespace std;
int k;
void perm( string s, int i, int n, int count){
   if( count == k ) {
      cout<< s.substr( i-k, k) <<endl;
      return;
   }
   if( i == n ) {
      return;
   }
   for( int a = i; a<n; a++ ) {
         swap(s[i], s[a]);
           perm(s, i+1, n, count+1);
           swap(s[i], s[a]);
   }
}
int main() {
   string s;
   cout << "Enter the string : " << endl ;
   cin >> s;
   cout << "Enter the string size" << endl ;
   cin>>k ;
   perm(s, 0, s.size(), 0);
   return 0;
}
```

Explanation :

Here , maximum concept is similar with task-1 . In this program , user can decide the size of the string even after entering the string . If the user has given “abc” as a input ,but he wants to reduce the size of the input , then he can enter the size and the compiler will work according to it . The permutation function counts the size and it uses a **substring** function which helps to create all possible permutation.

Sample Input :

abc
and k = 2

Sample output :
ab
ac
bc