Programs

P_0 Count the length of the string

```c
#include<stdio.h>

// Calculating the length of the characters in the string
int main()
{
   char name[] ="Sheldon Cooper";
   int len=0;

   for (int i = 0; name[i]!='\0'; i++)
   {
       if(name[i]!=' ')len++; // this will skip the spaces
   }

   printf("The length of the string is : %d\n",len);



   return 0;
}
```

P_02 : Convert the uppercase string to the lowercase string

```c
#include<stdio.h>

int main()
{
   char A[]="SHELDON COOPER";

   printf("[+]Uppercase String : %s\n",A);

  for (int i = 0; A[i]!='\0'; i++)
  {
       if(A[i]!=' ')A[i] = A[i] + 32; // this will skip the space
  }

  printf("[+]Lowercase String : %s\n",A);

   return 0;
}
```

P_02 : Convert the lowercase string to uppercase

```c
#include<stdio.h>

int main()
{
   char name[] ="sheldon cooper";

   printf("Lowercase String :  %s\n",name);

   for (int i = 0; name[i]!='\0'; i++)
   {
       if(name[i]!=' ')name[i]-=32;   // This condition will skip the spaces
   }

   printf("Uppercase String :  %s\n",name);

   return 0;
}
```

P_04 : Toggle the case of the string

```c
#include<stdio.h>

int main()
{
   /* Toggling :
    *  if character is lowercase change it to the uppercase
    *  if character is uppercase change it to the lowercase
   */
   char A[]="wElCoMe";

   printf("Original String : %s\n",A);

   for (int i = 0; A[i]!='\0'; i++)
   {
       if(A[i] >= 65 && A[i] <= 90 && A[i]!=' ')
       {
           A[i]+=32;
       }else if(A[i]>=97 && A[i] <= 122 && A[i]!=' ')
       {
           A[i]-=32;
       }
   }
   printf("Toggled String : %s\n",A);
   return 0;
}
```

P_05 : Count Characters, Words, Vowels, & Consonants in a string

```c
#include<stdio.h>

int main()
{
   char A[] = "Its so funny, how most blessed ones are the most cursed ones";

   int wCout  = 0;
   int vCount = 0;
   int cCount = 0;
   int spaceCount=0;

   // Counting the vowels and consonants
   for (int i = 0; A[i]!='\0'; i++)
   {
       if(A[i]=='a'||A[i]=='e'||
A[i]=='i'||A[i]=='o'||A[i]=='u'||A[i]=='A'||A[i]=='E'||A[i]=='I'||A[i]=='O'||A[i]=='U'
)
       {
           vCount++;
       }else if((A[i]>=65 && A[i]<=90)|| (A[i]>=97 && A[i]<=122)){
           cCount++;
       }

       if(A[i]==' ' && A[i-1]!=' ')
       {
            spaceCount++;
       }
   }

   wCout = spaceCount+1; // Number of words equals number of whiteSpace + 1


   printf("[+]Vowels           :   %d\n",vCount);
   printf("[+]Consonant        :   %d\n",cCount);
   printf("[+]Words            :   %d\n",wCout);


   return 0;
}
```

P_06 : Validating the String

```c
#include<stdio.h>

int main()
{
   /**
    * Invalid string as it contains the special
    * symbol @
   */
   char A[] = "sheldon@123";
   char B[] ="sheldon_283_@12"; // try for this one
   int isValid = 1;

   for (int i = 0; A[i]!='\0'; i++)
   {
       if(!(A[i]>=65 && A[i]<=90) && !(A[i]>=97 && A[i]<=122) && !(A[i]>=48 &&
A[i]<=57))
       {
           isValid=0;
           break;
       }
   }

   if(isValid)
   {
       printf("%s is valid username\n",A);
   }
   else{
       printf("%s is not a valid username\n",A);
   }

   return 0;
}
```

P_07 : Reverse the string [Memory Consuming Method]

```c
#include<stdio.h>

void reverseString1()
{
   char A[] = "This Is Sample String";
   char B[25];
   int i,j;

   printf("Original string : %s\n",A);

   for ( i = 0; A[i]!='\0'; i++){}

   i = i-1; // This skips the NULL character

   for ( j = 0; i >=0; j++,i--)
   {
       B[j]=A[i];
   }

   B[j]='\0';

   printf("Reversed String : %s\n",B);

}

int main()
{
   reverseString1();
   return 0;
}
```

P_08 : Reverse the string [Memory Efficient Method]

```c
#include<stdio.h>

void reverseString2()
{
   char A[]="This is also sample string";
   char temp;
   int i,j;

   printf("Original String : %s\n",A);

   for (i = 0; A[i]!='\0'; i++){}
   i = i - 1;

   for ( j = 0;i>j; j++,i--)
   {
       temp = A[j];
       A[j] = A[i];
       A[i] = temp;
   }

   printf("Reversed String : %s\n",A);
}

int main()
{

   reverseString2();
   return 0;
}
```

P_09 : Compare two strings, are equal or not [Method-Version-1]

```c
#include<stdio.h>

void compareString(char S1[], char S2[])
{
   int isSame = 1;
   for (int i = 0, j=0; S1[i] != '\0'; i++,j++)
   {
       if(S1[i] != S2[j])
       {
           isSame = 0;
           break;
       }
   }

   if(isSame)
   {
       printf("Strings are same\n");
   }else{
       printf("Strings are not same\n");
   }

}

int main()
{

   return 0;
}
```

P_10 : Compare two strings, are equal or not [Method-Version-2]

```c
#include<stdio.h>

void compareString2(char S1[], char S2[])
{
  int i,j;

   for ( i = 0,j=0; S1[i]=='\0'; i++,j++)
   {
       if(S1[i]!=S2[j])
       {
           break;
       }
   }

   if(S1[i] == S2[j])
   {
       printf("Strings are equal.\n");
   }else if(S1[i] > S2[j])
   {
       printf("String-1 is greater than String-2.\n");
   }else{
       printf("String-2 is greater than String-1.\n");
   }
}

int main()
{
   char A[]="hello";
   char B[]="hello";
   compareString2(A,B);

   char C[]="it's so funny,how most blessed ones are the most cursed ones";
   char D[]="no one cares how you feel, it's all about making dollar bills"; // :-)
   compareString2(C,D);

   return 0;
}
```

P_11 : Comparing two strings are equal or not [Optimized Method]

```c
#include<stdio.h>

int HASH[26] = {0};

void compareString3(char A[], char B[])
{
   int i,j;

   for ( i = 0;A[i]!='\0'; i++)
   {
       HASH[A[i]-97]++;
   }

   for ( j = 0; B[j]!='\0'; j++)
   {
       HASH[B[j]-97]--;

       if(HASH[B[j]-97] < 0)
       {
           break;
       }

   }
   if(HASH[B[j]-97] < 0)
   {
       printf("Strings are not equal\n");
   }else{
       printf("Strings are equal\n");
   }
}

int main()
{
   char A[]="hello";
   char B[]="hello";
   compareString3(A,B); // Strings are equal s

   char C[] = "friend";
   char D[] = "stabber";
   compareString3(C,D);  // Strings are not equal;

   return 0;
}
```

P_12 : Checking the string is palindrome or not [Slower & Memory Consuming Method]

```c
#include<stdio.h>
// Easy way
void isPalimdrome1(char A[])
{
   int i,j;
   char B[255];

   for ( i = 0; A[i]!='\0'; i++){} // Reaching the end of string
   i = i - 1;      // Reaching to the last char.(Just before the end of string)

   // Copying the string A to B in reverse fashion
   for (j = 0; i >= 0; i--,j++)
   {
       B[j] = A[i];
   }
   B[j] = '\0';

   // Checking the strings, comparing them
   for (i = 0, j=0; A[i]!='\0' && B[j]!='\0'; i++,j++)
   {
       if(A[i]!= B[j]){break;}
   }

   if(A[i] == B[j])
   {
       printf("String is palindrome\n");
   }else{
       printf("String is not palindrome\n");
   }
}

int main()
{
   char A[] = "racecar";
   isPalimdrome1(A);

   char B[] = "friend";
   isPalimdrome1(B);

   char C[] = "lil";
   isPalimdrome1(C);

   char D[] ="madam";
   isPalimdrome1(D);
   return 0;
}
```

P_13 : Checking the string is palindrome or not [Better Method]

```c
#include<stdio.h>
// Faster Method
void isPlaindrome2(char A[])
{
  int i,j;
  int isPalindrome = 1;

  //Traversing till the end of the string
  for ( j = 0; A[j]!='\0'; j++){}
  j = j-1; // Reaching the last character

   for (i = 0; j>i; i++,j--)
   {
       if(A[i] != A[j])
       {
           isPalindrome = 0;
           break;
       }
   }

   if(isPalindrome)
   {
       printf("String is palindrome\n");
   }else{
       printf("String is not palindrome\n");
   }

}

int main()
{
   char A[]="racecar";
   isPlaindrome2(A);
   return 0;
}
```

P_14 : Checking and counting the duplicates in the String

```c
#include<stdio.h>

/**
* Slower Method
*/
void findDupliactes1(char A[])
{
   int count = 0;
   for (int i = 0; A[i+1]!='\0'; i++)
   {
       if(A[i]!= -1 && A[i] !=' ') // tacking the spaces
       {
           count = 1;
           for (int j = i+1;A[j]!='\0'; j++)
           {
               if(A[i] == A[j])
               {
                   count++;
                   A[j] = -1; // Marking the duplicate elements as -1
               }
           }

           if(count > 1)
           {
               printf("Character %c is occured %d times\n",A[i],count);
           }

       }

   }

}

int main()
{

   char A[] = "drain out bad energy, forget bad memories";
   findDupliactes1(A);
   return 0;
}
```

P_15 : Counting the duplicates in the string [Optimized Method]

```c
#include <stdio.h>

void findDuplicates2(char S[])
{
   int HASH[26] = {0};

   for (int i = 0; S[i] != '\0'; i++)
   {
           HASH[S[i] - 97]++;
   }

   for (int i = 0; i < sizeof(HASH) / sizeof(HASH[0]); i++)
   {
       if (HASH[i] > 1)
       {
           printf("Character %c is occurred %d times\n", i + 97, HASH[i]);
       }
   }
}

int main()
{
   char A[]="lets take vacation from the earth";
   findDuplicates2(A);
   return 0;
}
```

P_16 : Checking the duplicates in the string [Efficient Method]

```c
#include<stdio.h>

/**
* Bit-Set method : Finding Duplicates by setting individual bits in memory
* with the help :
*  1. Masking Operation (ANDing)
*  2. Merging Operation  (ORing)
*  3. Bitwise Shifting Operation (Left Shifting)
*
* Faster Method than previous 2 methods
* Most recommended method in terms of
*      [*] Memory efficiency
*      [*] Execution Speed
*
*/

void isContainDuplicates(char S[])
{
   int i;
   int A = 0; // 0000 0000 0000 0000 0000 0000 0000 0000
   int H = 0; // 0000 0000 0000 0000 0000 0000 0000 0000

   for ( i = 0; S[i]!='\0'; i++)
   {
       A = 1; // 0000 0000 0000 0000 0000 0000 0000 0001
       A = A << (S[i]-97);

       if((H & A) > 0)
       {
           printf("%c is duplicate \n",S[i]);
       }else{
           H = A | H;
       }
   }

}

int main()
{
   char A[] = "findings";
   isContainDuplicates(A);
   return 0;
}
```

P_17 : Checking if the strings are anagram or not [Normal Method]

```c
#include<stdio.h>

/**
* Slower Method
* This methods is only applicable for the string containing
* the unique characters
*/
void isAnagram1(char A[], char B[], int lenA, int lenB)
{
   int i;
   int j;
   int isAnagram = 0;

   for ( i = 0; A[i]!='\0' && lenB == lenA; i++)
   {
       isAnagram = 0;
       for ( j = 0; B[j]!='\0'; j++)
       {
           if(A[i] == B[j])
           {
               isAnagram = 1;
               break;
           }
       }

       if(!isAnagram){break;}
   }

   if(isAnagram)
   {
       printf("Two string are anagram...\n");
   }else{
       printf("Two strings are not anagram...\n");
   }

}

int main()
{
   char A[]="cooper";
   char B[]="poocer";

   int lenA = sizeof(A)/sizeof(A[0]);
   int lenB = sizeof(B)/sizeof(B[0]);
```

```
   isAnagram1(A,B,lenA,lenB);


   char C[] = "coper";
   char D[] = "pocer";

   isAnagram1(C,D,sizeof(C)/sizeof(C[0]),sizeof(D)/sizeof(D[0]));

   return 0;
}
```

P_18 : Checking if the strings are anagram or not with duplicates [Better Method]

```
#include<stdio.h>

void isAnagram2(char A[], char B[],int lenA, int lenB)
{
   int isAnagram = 0;

   for (int i = 0;A[i]!='\0'; i++)
   {
       isAnagram = 0;
       for (int j=0;B[j]!='\0'; j++)
       {
           if(A[i] == B[j])
           {
               isAnagram = 1;
               B[j] = -1; // this will mark the elements to -1, to handle
duplicates
               break;
           }
       }
       if(!isAnagram){break;}
   }

   if(isAnagram)
   {
       printf("Strings are anagram\n");
   }else{
       printf("String are not anagram\n");
   }
```

```
}

int main()
{
   char A[] = "cooper";
   char B[] = "poocer";
   int lenA = sizeof(A);
   int lenB = sizeof(B);
   isAnagram2(A,B,lenA,lenB);

   char R[] = "pranay";
   char S[] = "jarred";
   int lenR = sizeof(R);
   int lenS = sizeof(S);
   isAnagram2(R,S,lenR,lenS);

   return 0;
}
```

P_19 : Checking if the strings are anagram or not with duplicates [Optimal Method]

```
#include<stdio.h>

void isAnagram3(char A[], char B[],int lenA,int lenB)
{
   /**
    * Moderately faster method than two previous methods
    * The major drawback of this method is, it consumes
    * extra memory size for hash-table
   */
   int HASH[25] = {0}; // Hash Table
   int i,j;

   if(lenA != lenB)
   {
       printf("Strings are not anagram\n");
       return;
```

```
    }
    for (i = 0;A[i]!='\0'; i++)
    {
        HASH[A[i]-97]++;
    }

    for (j = 0; B[j]!='\0'; j++)
    {
        HASH[B[j]-97]--;
        if(HASH[B[j]-97] < 0){break;}
    }

    if(HASH[B[j]-97] < 0)
    {
        printf("Strings are not anagram\n");
    }else{
        printf("Strings are anagram\n");
    }


}

int main()
{
    char A[] = "cooper";
    char B[] = "poocer";


    int lenA = sizeof(A)/sizeof(A[0]);
    int lenB = sizeof(B)/sizeof(B[0]);

    isAnagram3(A,B,lenA,lenB);

    return 0;
}
```

P_20 : Checking if the strings are anagram or not with duplicates [Efficient Method]

```c
#include<stdio.h>

void isAnagram4(char A[], char B[])
{
   int HASH = 0; // 0000 0000 0000 0000 0000 0000 0000 0000
   int H = 0;    //  0000 0000 0000 0000 0000 0000 0000 0000

   for (int i = 0; A[i]!='\0'; i++)
   {
       H = 1;   // 0000 0000 0000 0000 0000 0000 0000 0001
       H = H << (A[i]-97);
       HASH = HASH | H; // ORing (Merging Operation)
   }

   for (int i = 0; B[i]!='\0'; i++)
   {
       H = 1;
       H = H << B[i]-97;
       if((HASH & H) < 1){break;} // ANDing (Masking Opertion )
   }

   if((HASH & H) > 0)
   {
       printf("Strings are anagram\n");
   }else{
       printf("Strings are not anagram\n");
   }



}
int main()
{
   char A[]="decomal";
   char B[]="medical";

   isAnagram4(A,B);

   return 0;
}
```

P_21 : Print the permutations of the string

```c
#include<stdio.h>

void perm(char S[], int k)
{
   static char R[10] = {0};
   static int A[10] = {0};

   if(S[k] == '\0')
   {
       R[k] = '\0';
       printf("%s\n",R);
   }else{
       for (int i = 0; S[i]!='\0'; i++)
       {
           if(A[i] == 0)
           {
               A[i] = 1;
               R[k] = S[i];
               perm(S,k+1);
               A[i] = 0;
           }
       }

   }
}

int main()
{
   char S[]="ABC";
   perm(S,0);
   return 0;
}
```