ASSIGNMENT

Strlen()

```c
#include<stdio.h>
#include<string.h>

int main(){
    char name[]="Soumya";
    printf("Length of name : %ld\n",strlen(name));
    return 0;
}
```

Output:

```
Length of name : 6
```

Strcpy()

```c
#include<stdio.h>
#include<string.h>
int main(){
    char str1[10];
    char str2[10];
    strcpy(str1,"Soumya");
    strcpy(str2,"Mariyam");
    printf("str1[] = %s\tstr2[]=%s",str1,str2);
    return 0;
}
```

Output:

```
str1[] = Soumya str2[]=Mariyam
```

Strncpy()

```c
#include<stdio.h>
#include<string.h>
int main(){
   char str1[10];
   char str2[10];
   strcpy(str1,"Soumya");
   strncpy(str2,str1,4);
   printf("str1[]=%s \t str2[]=%s",str1,str2);
   return 0;
}
```

Strcat()

```c
#include<stdio.h>
#include<string.h>
int main(){
   char str1[10];
   char str2[10];
```

```
    strcpy(str1,"Soumya");
    strncpy(str2,str1,4);
    printf("str1[]=%s \t str2[]=%s\n",str1,str2);
    strcat(str1,str2);
    printf("str1[]=%s\t str2[]=%s\n",str1,str2);
    return 0;
}
```

<u>Strcmp()&Strcnp() /Comparing Strings</u>

```
#include<stdio.h>
#include<string.h>
int main(){
    char A[10]="soumya";
    char B[10]="mariyam";
    printf("strcmp(\"A\",\"A\")is "); //comparing same character we get output 0;
    printf("%d\n",strcmp("A","A"));
    printf("strcmp(\"A\",\"B\")is "); //comparing different character we get output -1;
    printf("%d\n",strcmp("A","B"));
    printf("strcmp(\"C\",\"A\")is "); //comparing different character we get output 1;
    printf("%d\n",strcmp("C","A"));
    printf("strcmp(\"A\",\"D\")is "); //comparing different character we get output -1;
    printf("%d\n",strcmp("A","D"));
    printf("strcmp(\"D\",\"A\")is "); //comparing different character we get output -1;
    printf("%d\n",strcmp("D","A"));
    printf("strcmp(\"apples\",\"apple\")is ");
    printf("%d\n",strcmp("apples","apple"));
    char str1[10]="ABCD";
    char str2[10]="ABBD";
    printf("strcmp(\"str1\",\"str2\")is ");
    printf("%d\n",strcmp("str1","str2"));
    printf("strcmp(\"Astounding\",\"Astso\")is ");
    printf("%d\n",strncmp("Astounding","Astso",5));

}
```

Output:

```
strcmp("A","A")is 0
strcmp("A","B")is -1
strcmp("C","A")is 1
strcmp("A","D")is -1
strcmp("D","A")is 1
strcmp("apples","apple")is 1
strcmp("str1","str2")is -1
strcmp("Astounding","Astso")is -4
```

<u>Strchr() /Searching of a single character</u>

```
#include<stdio.h>
#include<string.h>

int main(){
    char str[]="Hi my name is Soumya";
    int l=strlen(str);
    for(int i=0;i<l;i++){
        printf("str[%d] = %c,address= %p\n",i,str[i],(str+i));
    }
    char ch='n';
    char *pFound=NULL;
    pFound=strchr(str,ch);
    printf("pFound = %p",pFound);

    return 0;
}
```

Output:

```
str[6] = n,address= 0061FF01
pFound = 0061FF01
```

Strstr()/Searching a word

```
#include<stdio.h>
#include<string.h>

int main(){
    char text[]="Every dog has his day";
    char word[]="dog";
    int length_word=strlen(word);
    char *pFound=NULL;
    pFound=strstr(text,word);
    printf("The found string : %.*s\n",length_word,pFound);//The %. *s format specifier is used to print only a certain number of characters from the found string.
    printf("pFound = %p",pFound);
}
```

Output:

```
The found string : dog
pFound = 0061FF08
```

Strtok()/Tokenizing a String

```
#include<stdio.h>
#include<string.h>

int main(){
    char str[]="Hi my - name is - Soumya";
    char s[2]="-";
    char *token=NULL;
    token=strtok(str,s);

    while(token !='\0'){
        printf("Token = %s\n",token);
        token=strtok(NULL,s);
    }
    return 0;
}
```

Output:

```
Token = Hi my
Token =  name is
Token =  Soumya
```

Analyzing Strings

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main(){
    char buff[100];
    int nLetters=0;
    int nDigits=0;
    int nPunct=0;
    printf("Enter an interesting string of less than %d characters:\n",100);
    scanf("%[^\n]",buff);
    int i=0;
    while(buff[i]){
        if(isalpha(buff[i]))
            ++nLetters;
        else if(isdigit(buff[i]))
            ++nDigits;
        else if(ispunct(buff[i]))
            ++nPunct;
        ++i;
    }
    printf("\n Your string contained %d letters,%d digits and %d punctuation
characters.\n",nLetters,nDigits,nPunct);
}
```

Output:

```
Enter an interesting string of less than 100 characters:
I am soumya ,and i am 23.
 Your string contained 15 letters,2 digits and 2 punctuation characters.
```

Converting Strings

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>

int main(){
    char text[100];
    char substring[40];
    printf("Enter the string to be seacrched (less than %d characters):\n",100);
    scanf("%[^\n]",text);
    printf("\nEnter the string sought (less than %d characters):\n",40);
    scanf("%s",substring);
    printf("\nFirst string entered :\n%s\n",text);
    printf("Second string entered:\n%s\n",substring);
    for(int i=0;(text[i]=(char)toupper(text[i]))!='\0';++i);
    for(int i=0;(substring[i]=(char)toupper(substring[i]))!='\0';++i);
    printf("The second string %s found in  the first.\n",((strstr(text,substring)==NULL)?"was
not":"was"));
}
```

Output:

```
First string entered :
every dog has a day
Second string entered:
dog
The second string was found in  the first.
```

```c
#include <stdio.h>

void copyStringArray(char to[], char from[]);
void copyStringPointer(char *to, char *from);
int main() {
    char A[20]={};
    char B[20];
    int choice;
    printf("Enter a string for B (less than 20 characters): ");
    scanf(" %[^\n]", B);
    printf("\nChoose the method to copy the string:\n");
    printf("1. Array notation\n");
    printf("2. Pointer notation\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            copyStringArray(A, B);
            printf("String in A after copy (array version): %s\n", A);
            break;
        case 2:
            copyStringPointer(A, B);
            printf("String in A after copy (pointer version): %s\n", A);
            break;
        default:
            printf("Invalid choice.\n");
            return 1;
    }
    return 0;
}
void copyStringArray(char to[], char from[]) {
    int i;
    for (i = 0; from[i] != '\0'; ++i) {
        to[i] = from[i];
    }
    to[i] = '\0';
}
void copyStringPointer(char *to, char *from) {
    for (; *from != '\0'; ++from, ++to) {
        *to = *from;
    }
    *to = '\0';
}
```

```
Enter a string for B (less than 20 characters): soumya

Choose the method to copy the string:
1. Array notation
2. Pointer notation
Enter your choice: 2
String in A after copy (pointer version): soumya

Choose the method to copy the string:
1. Array notation
2. Pointer notation
Enter your choice: 1
String in A after copy (array version): soumya
```

Problem 1: Palindrome Checker
Problem Statement:
Write a C program to check if a given string is a palindrome. A string is considered a palindrome if it reads the same backward as forward, ignoring case and non-alphanumeric characters. Use functions like strlen(), tolower(), and isalpha().
Example:
Input: "A man, a plan, a canal, Panama"
Output: "Palindrome"

Malloc

```c
#include<stdio.h>
#include<stdlib.h>

int main(){
    int *ptr;
    int num,i;
    printf("Enter the number of elements :");
    scanf("%d",&num);
    printf("\n");
    printf("The number entered is n = %d \n",num);

    //Dynamically Allocating Memory for the array;
    ptr=(int *)malloc(num * sizeof(int));
    //Check whether the memory is allocated successfully or not;
    if(ptr==NULL){
        printf("Memory not allocated \n");
        exit(0);
    }
    else{
        printf("Memory is allocated successfully \n");
    }
    //Populating the array;
    for(i=0;i<num;i++){
        ptr[i]=i+1;
    }
    //Displaying the array;
    for(i=0;i<num;i++){
        printf("%d,",ptr[i]);
    }
    //Free the Dynamically allocated memory;
    free(ptr);
    return 0;
}
```

Output:

```
The number entered is n = 6
Memory is allocated successfully
1,2,3,4,5,6,
```

Problem 1: Palindrome Checker
Problem Statement:
Write a C program to check if a given string is a palindrome. A string is considered a palindrome if it reads the same backward as forward, ignoring case and non-alphanumeric characters. Use functions like strlen(), tolower(), and isalpha().
Example:
Input: "A man, a plan, a canal, Panama"
Output: "Palindrome"

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main(){
    char str[50];
    printf("Enter the Input :");
    scanf("%d[^\n]",str);
    int l=strlen(str),j=l-1,i;
    for(i=0;i<j;){
        if(!isalpha(str[i])){
            i++;
            continue;
        }
        if(!isalpha(str[j])){
            j--;
            continue;
        }
        if(tolower(str[i])!=tolower(str[j])){
            printf("Not Palindrome\n");
            return 0;
        }
        i++;
        j--;
    }
    printf("Palindrome\n");
}
```

Output

```
Enter the Input :malayalam
Palindrome
```

Problem 2: Word Frequency Counter

Problem Statement:

Write a program to count the frequency of each word in a given string. Use strtok() to tokenize the string and strcmp() to compare words. Ignore case differences.

Example:

Input: "This is a test. This test is simple."

Output:

Word: This, Frequency: 2

Word: is, Frequency: 2

Word: a, Frequency: 1

Word: test, Frequency: 2

Word: simple, Frequency: 1

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char str[200];
    printf("Enter a string: ");
    scanf("%[^\n]",str);

    for(int i = 0; str[i]; i++) {    // Convert the entire string to lowercase
        str[i] = tolower(str[i]);
    }
    char temp[200];
    strcpy(temp, str);
    char *words[100];      // Tokenize the string and store words in an array
    int word_count[100] = {0};
    int total_words = 0;
    char *token = strtok(temp, " .");
    while (token != NULL) {
        int found = 0;
        for (int i = 0; i < total_words; i++) {
            if (strcmp(words[i], token) == 0) {
                word_count[i]++;
                found = 1;
                break;
            }
        }
        if (!found) {
            words[total_words] = token;
            word_count[total_words]++;
            total_words++;
        }
        token = strtok(NULL, " .");
    }
    printf("Word Frequencies:\n");
    for (int i = 0; i < total_words; i++) {
        printf("Word: %s, Frequency: %d\n", words[i], word_count[i]);
    }
    return 0;
}
```

Output:

```
Enter a string: i am happy .i am happy and good
Word Frequencies:
Word: i, Frequency: 2
Word: am, Frequency: 2
Word: happy, Frequency: 2
Word: and, Frequency: 1
Word: good, Frequency: 1
```

Problem 3: Find and Replace

Problem Statement:

Create a program that replaces all occurrences of a target substring with another substring in a given string. Use strstr() to locate the target substring and strcpy() or strncpy() for modifications.

Example:

Input:

String: "hello world, hello everyone"

Target: "hello"

Replace with: "hi"

Output: "hi world, hi everyone"

```c

#include <stdio.h>
#include <string.h>
void replacement(char *str, const char *target, const char *replace);
int main() {
    char str[200];
    char target[50];
    char replace[50];
    printf("Enter the string: ");
    scanf("%[^\n]", str);
    getchar();
    printf("Enter the target string: ");
    scanf("%[^\n]", target);
    getchar();
    printf("Enter the replace string: ");
    scanf("%[^\n]", replace);
    getchar();
    replacement(str, target, replace);
    printf("Modified string is: %s\n", str);
    return 0;
}
void replacement(char *str, const char *target, const char *replace) {
    char result[200];
    char *pos;
    int target_len = strlen(target);
    int replace_len = strlen(replace);
    int index = 0;
    result[0] = '\0';
    while ((pos = strstr(str, target)) != NULL) {

        int len = pos - str;
        strncat(result, str, len);
        strcat(result, replace);

        str = pos + target_len;
    }
    strcat(result, str);

    strcpy(str, result);
}
```

Problem 4: Reverse Words in a Sentence
Problem Statement:
Write a program to reverse the words in a given sentence. Use strtok() to extract words and strcat() to rebuild the reversed string.
Example:
Input: "The quick brown fox"
Output: "fox brown quick The"

```c
#include <stdio.h>
#include <string.h>

void reverseWords(const char *sentence, char *reversed);
int main() {
    char sentence[200];
    char reversed[200] = "";
    printf("Enter a sentence: ");
    scanf("%[^\n]",sentence);
    reverseWords(sentence, reversed);
    printf("Reversed sentence: %s\n", reversed);
    return 0;
}
void reverseWords(const char *sentence, char *reversed) {
    char temp[200];
    strcpy(temp, sentence);
    char *words[100];
    int count = 0;
    char *token = strtok(temp, " ");
    while (token != NULL) {
        words[count++] = token;
        token = strtok(NULL, " ");
    }
    for (int i = count - 1; i >= 0; i--) {
        strcat(reversed, words[i]);
        if (i > 0) {
            strcat(reversed, " ");
        }
    }
}
```

Output:

```
Enter a sentence: i am sou
Reversed sentence: sou am I
```

Problem 5: Longest Repeating Substring

Problem Statement:

Write a program to find the longest substring that appears more than once in a given string. Use strncpy() to extract substrings and strcmp() to compare them.

Example:

Input: "banana"

Output: "ana"

```c
#include <stdio.h>
#include <string.h>

void longestRepeatingSubstring(const char *str, char *result);
int main() {
    char str[100];
    char result[100] = "";
    printf("Enter the string: ");
    scanf("%[^\n]",str);
    longestRepeatingSubstring(str, result);
    if (strlen(result) > 0) {
        printf("Longest repeating substring: %s\n", result);
    } else {
        printf("No repeating substring found.\n");
    }
    return 0;
}
void longestRepeatingSubstring(const char *str, char *result) {
    int len = strlen(str);
    int maxLen = 0;
    for (int i = 0; i < len; i++) {
        for (int j = i + 1; j < len; j++) {
            int k = 0;
            while (i + k < len && j + k < len && str[i + k] == str[j + k]) {
                k++;
            }
            if (k > maxLen) {
                maxLen = k;
                strncpy(result, &str[i], k);
                result[k] = '\0';  // Null-terminate the result
            }
        }
    }
}
```

Output:

```
Enter the string: banana
Longest repeating substring: ana
```