String Functions

Strlen

```
#include<stdio.h>
#include<string.h>
int main(){
    char str1[] = "Anjali Lal";
    printf("The length of str1 = %ld\n",strlen(str1));

    return 0;
}
```

Strcpy

```
#include<stdio.h>
#include<string.h>
int main(){
    char str1[10];
    char str2[10];
    strcpy(str1,"Anjali");
    strcpy(str2,"Lal");
    printf("str1[] = %s \t str2[] = %s",str1,str2);

    return 0;
}
```

Strncpy

```
#include<stdio.h>
#include<string.h>
int main(){
    char str1[10];
    char str2[10];
    strcpy(str1,"Anjali");
    strncpy(str2,str1,5);
    printf("str1[] = %s \t str2[] = %s",str1,str2);
    return 0;
}
```

<u>strcat</u>

```
#include<stdio.h>
#include<string.h>
int main(){
    char str1[10];
    char str2[10];
    strcpy(str1,"Anjali");
    strcpy(str2,"Lal");
    strcat(str1,str2);
    printf("str1[] = %s \t str2[] = %s",str1,str2);

    return 0;
}
```

Strcmp

```
#include<stdio.h>
#include<string.h>
int main(){
    printf("strcmp(\"A\",\"A\") is ");
    printf("%d\n",strcmp("A","A"));

    printf("strcmp(\"A\",\"B\") is ");
    printf("%d\n",strcmp("A","B"));

    printf("strcmp(\"B\",\"A\") is ");
    printf("%d\n",strcmp("B","A"));

    printf("strcmp(\"C\",\"A\") is ");
    printf("%d\n",strcmp("C","A"));
    printf("strcmp(\"Z\",\"a\") is ");
    printf("%d\n",strcmp("Z","a"));

    printf("strcmp(\"apples\",\"apple\") is ");
    printf("%d\n",strcmp("apples","apple"));

    printf("%d\n",strcmp("ABCD","ABBD"));
    return 0;
}
```

<u>Strncmp</u>

```
#include<stdio.h>
#include<string.h>
int main(){
    printf("strcmp(\"Astounding\",\"Astro\") is ");
    printf("%d\n",strcmp("Astounding","Astro"));

    printf("strncmp(\"Astounding\",\"Astro\") is ");
    printf("%d\n",strncmp("Astounding","Astro",5));

    return 0;
}
```

<u>Strchr()</u>

```
#include<stdio.h>
#include<string.h>
int main(){
    char str[] = "Hi my name is Anjali";
    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;
}
```

<u>Output</u>

```
str[0] = H, address = 0x7ffc4ff50010
str[1] = i, address = 0x7ffc4ff50011
str[2] =  , address = 0x7ffc4ff50012
str[3] = m, address = 0x7ffc4ff50013
str[4] = y, address = 0x7ffc4ff50014
str[5] =  , address = 0x7ffc4ff50015
str[6] = n, address = 0x7ffc4ff50016
str[7] = a, address = 0x7ffc4ff50017
str[8] = m, address = 0x7ffc4ff50018
str[9] = e, address = 0x7ffc4ff50019
str[10] =  , address = 0x7ffc4ff5001a
str[11] = i, address = 0x7ffc4ff5001b
str[12] = s, address = 0x7ffc4ff5001c
str[13] =  , address = 0x7ffc4ff5001d
str[14] = A, address = 0x7ffc4ff5001e
str[15] = n, address = 0x7ffc4ff5001f
str[16] = j, address = 0x7ffc4ff50020
str[17] = a, address = 0x7ffc4ff50021
str[18] = l, address = 0x7ffc4ff50022
str[19] = i, address = 0x7ffc4ff50023
pFound = 0x7ffc4ff50016
```

strstr()

```
#include<stdio.h>

#include<string.h>

int main(){

   char text[] = "Every dog has his day";

   int l = strlen(text);

   for(int i=0;i<l;i++){

      printf("text[%d] = %c, address = %p\n",i,text[i],(text+i));

   }

   char word[]="dog";

   char *pFound = NULL;

   pFound = strstr(text,word);

   printf("pFound = %p",pFound);


   return 0;

}
```

Strtok()

```
#include<stdio.h>

#include<string.h>

int main(){

   char str[] = "Hi my - name is - Anjali";

   char s[2] = "-";

   char *token = NULL;

   token = strtok(str,s);


   while(token != '\0'){

      printf("token = %s\n",token);

      token = strtok(NULL,s);

   }

   return 0;

}
```

Converting strings

```
#include<stdio.h>

#include<string.h>

int main(){

   char text[100];

   char substring[40];

   printf("Enter the string to searched: \n");

   scanf("%s",text);

   printf("Enter the string sought: \n");

   scanf("%s",substring);

   printf("First string entered: %s\n",text);

   printf("Second string entered: %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");

   return 0;

}


#include<stdio.h>

#include<string.h>

void copyArray(char A[],char B[]);

void copyPointer(char *A,char *B);

int main(){

   char A[20];

   char B[20];

   char op;

   printf("Enter your name: \n");

   scanf("%s",B);

   printf("Enter your option 'a' for array notation and 'p' for pointer notation: \n");

   scanf(" %c",&op);
```

```
    switch(op){
        case 'a':
            copyArray(A,B);
            break;
        case 'p':
            copyPointer(A,B);
            break;
        default:
            printf("Invalid option!\n");
            break;
    }
    return 0;
}
void copyArray(char A[],char B[]){
    int i=0;
    for(i=0;B[i]!='\0';i++){
        A[i]=B[i];
    }
    A[i]='\0';
    printf("Copied string: %s\n",A);
}
void copyPointer(char *A,char *B){
    while (*B != '\0') {
        *A = *B;
        A++;
        B++;
    }
    *A = '\0';
    printf("Copied string: %s\n", A);
}
```

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"

```
 #include<stdio.h>

#include<string.h>


int palindrome(char str[]);

int main(){

   char str[100];

   printf("Enter the string: \n");

   scanf("%s",str);

   if(palindrome){

      printf("Palindome");

   }else{

      printf("Not palindrome");

   }

   return 0;

}

int palindrome(char str[]){

   int start = 0;

   int end = strlen(str)-1;

   if(!isalnum(str[start])){

      start++;

   }else if(!isalnum(str[end])){

      end--;

   }else if(tolower(str[start]!=tolower(str[end]))){

      return 0;

   }
```

```
    else{
        start++;
        end--;
    }
    return 1;
}
```

==

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
```

==

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"

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


void replacement(char *str, const char *target, const char *replace);


int main() {
    char str[100];
    char target[20];
    char replace[20];
    printf("Enter the string: \n");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = 0;

    printf("Enter the target string: \n");
    fgets(target, sizeof(target), stdin);
    target[strcspn(target, "\n")] = 0;

    printf("Enter the replace string: \n");
    fgets(replace, sizeof(replace), stdin);
    replace[strcspn(replace, "\n")] = 0;

    replacement(str, target, replace);

    printf("Modified string is: %s", str);

    return 0;
}

void replacement(char *str, const char *target, const char *replace) {
    char *p = strstr(str, target);
    int tl = strlen(target);
    int rl = strlen(replace);
    while (p != NULL) {
        char temp[100];
        int i = 0;
        while (str != p) {
            temp[i++] = *str++;
        }
        for (int j = 0; j < rl; j++) {
```

```
        temp[i++] = replace[j];
      }
      str = p + tl;
      while (*str) {
        temp[i++] = *str++;
      }
      temp[i] = '\0';
      strcpy(p - (str - p - tl), temp);
      p = strstr(str, target);
    }
}
```
===
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"

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[100];
    char *words[50];
    char reversed[100] = "";
    int word_count = 0;
    printf("Enter a sentence: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = 0;
    char *word = strtok(str, " ");
    while (word != NULL) {
        words[word_count++] = word;
        word = strtok(NULL, " ");
    }
    for (int i = word_count - 1; i >= 0; i--) {
        if (i != word_count - 1) {
            strcat(reversed, " ");
        }
        strcat(reversed, words[i]);
    }
    printf("Reversed sentence: %s\n", reversed);
    return 0;
}
```
==

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"

```
#include <stdio.h>
#include <string.h>
void findLongestRepeatingSubstring(char *str);
int main() {
    char str[100];
    printf("Enter the string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    findLongestRepeatingSubstring(str);
    return 0;
}
void findLongestRepeatingSubstring(char *str) {
    int len = strlen(str);
    char longestSubstring[100] = "";
    int longestLength = 0;

    for (int i = 0; i < len; i++) {

        for (int j = i + 1; j <= len; j++) {

            int subStrLength = j - i;

            if (subStrLength <= longestLength) {

                continue;

            }

            char subStr[100];

            strncpy(subStr, &str[i], subStrLength);

            subStr[subStrLength] = '\0';

            for (int k = 0; k < len - subStrLength + 1; k++) {

                if (k != i && strncmp(&str[k], subStr, subStrLength) == 0) {

                    if (subStrLength > longestLength) {

                        longestLength = subStrLength;

                        strcpy(longestSubstring, subStr);

                    }

                    break;

                }

            }

        }

    }

    if (longestLength > 0) {

        printf("Longest repeating substring: %s\n", longestSubstring);

    } else {

        printf("No repeating substrings found.\n");

    }}
```
==

<u>DYNAMIC MEMORY ALLOCATION</u>

<u>malloc()</u>

```
#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 allocate memory for the array
   ptr = (int *)malloc(num * sizeof(int));
   //Check wheter 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 alocated memory
   free(ptr);
   return 0;
   }
```