DAY 11

1) strlen prgm

```
#include <stdio.h>
#include<string.h>
int main()
{
    char str1[]="my string";
    int x=strlen(str1);
    printf("Length=%d",x);
    return 0;
}
```

2)strcpy

```
#include <stdio.h>
#include<string.h>
int main()
{
   char str1[]="navya";
   char str2[20];
  // strcpy(str2,str1);
  strcpy(str2,"ram");
   printf("string 2: %s",str2);
    return 0;
}
```

3)strncpy

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

```
   char str2[20];
   strcpy(str1,"navya");
   strncpy(str2,"malayalam",5);

   printf("str1[]:%s\nstr2[]: %s",str1,str2);
    return 0;
}

4)
#include <stdio.h>
#include<string.h>
int main()
{
   char str1[10];
   char str2[20];
   strcpy(str1,"hello");
   strncpy(str2,"ram",5);
   strcat(str1,str2);
   //printf("str1[]:%s\nstr2[]: %s",str1,str2);
   printf("string conatenation:%s\n",str1);
   printf("str1[]:%s\nstr2[]: %s",str1,str2);
    return 0;
}
5)
#include <stdio.h>
#include<string.h>
int main()
{
        char str1[10]="navya";
        char str2[20]="T S";
        //strcpy(str1,"hello");
        //strncpy(str2,"ram",5);
        //strcat(str1,str2);
        //printf("str1[]:%s\nstr2[]: %s",str1,str2);
        //printf("string conatenation:%s\n",str1);
        printf("str1[]:%s\nstr2[]: %s\n",str1,str2);
        if(strcmp(str1,str2)==0) {
```

```
		printf("equal");
	}
	else {
		printf("not equal");
	}
	printf("\nstrcmp(\"A\",\"A\")is:");
	printf("%d\n",strcmp("A","A"));

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

		printf("\nstrcmp(\"A\",\"C\")is:");
	printf("%d\n",strcmp("A","C"));

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


		printf("\nstrcmp(\"Z\",\"a\")is:");
	printf("%d\n",strcmp("Z","a"));


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

	return 0;
}
```

Output

```
[?2004l
str1[]:navya
str2[]: T S
not equal
strcmp("A","A")is:0

strcmp("A","B")is:-1
```

strcmp("A","C")is:-1

strcmp("B","A")is:1

strcmp("Z","a")is:-1

strcmp("apples","apple")is:-1
[?2004h

6)

```
#include <stdio.h>
#include<string.h>
int main()
{
        char str[]="hi ram";
        for(int i=0;i<strlen(str);i++){
            printf("str[%d]=%c,str[%d]=%p\n",i,str[i],i,(str+i));
        }
        char c='r';
        char *p=NULL;
        p=strchr(str,c);
        printf("p=%c\n",*p);
        printf("p=%p",p);
        return 0;
}
```

```
[?2004l
str[0]=h,str[0]=0x7ffcf7621021
str[1]=i,str[1]=0x7ffcf7621022
str[2]= ,str[2]=0x7ffcf7621023
str[3]=r,str[3]=0x7ffcf7621024
str[4]=a,str[4]=0x7ffcf7621025
str[5]=m,str[5]=0x7ffcf7621026
p=r
p=0x7ffcf7621024[?2004h
```

7)

```
#include <stdio.h>
#include<string.h>
int main()
{
        char str[]="hi ram";
        char word[]="ram";
        char *p=NULL;
        p=strstr(str,word);


        if(p!=NULL){
            printf("p=%c\n",*p);
            printf("address of p=%p\n",p);
            for(int i=0;i<strlen(word);i++){
                printf("%c",word[i]);
            }
        }else{
            printf("substring not found");
        }
        return 0;
}
```

8)

```
#include <stdio.h>
#include<string.h>
int main()
{
    char str[]="hi & hello,bye";
    char s[4]=",&";
    char *p=NULL;

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

4)

```
#include<stdio.h>
#include<string.h>
int main(){
    char string[100];
    char *arr[100];
    char reverse[100];

    printf("enter the sentence");
    scanf("%[^\n]",string);
    int inv=0;
    char *token =strtok(string," ");
    while(token!=NULL){
        arr[inv++]=token;
        token = strtok(NULL," ");

    }
```

```
    for (int i = inv - 1; i >= 0; i--) {
        strcat(reverse, arr[i]);
        if (i > 0) {
            strcat(reverse, " ");
        }
    }
    printf("Reversed sentence: %s\n", reverse);
}
```

9)

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main(){
    char str1[30];
    char str2[30];

    printf("enter  string :");
    fgets(str1,sizeof(str1),stdin);
    printf("enter second string to sought:");
    fgets(str2,sizeof(str2),stdin);


    for(int i=0;i<strlen(str1);i++){
        printf("%c",toupper(str1[i]));
    }

    for(int i=0;i<strlen(str2);i++){
        printf("%c",toupper(str2[i]));
    }
```

```
    printf("the second string %s found in
first\n",(strstr(str1,str2)==NULL?"was not":"was"));


    return 0;
}
```

10)

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

void copystring_array(char [], char []);
void copystring_pointer(char *, char *);

int main() {
    char str1[30];
    char str2[30];
    char choice;

    printf("Enter string: ");
    fgets(str1, sizeof(str1), stdin);

    // Remove the newline character left by fgets
    str1[strcspn(str1, "\n")] = '\0';

    printf("Enter choice ('p' for array, 's' for pointer): ");
    scanf("%c", &choice);  // Read user choice

    if (choice == 'p') {
        copystring_array(str1, str2);
        printf("str2[] = %s\n", str2);
    } else {
        copystring_pointer(str1, str2);
        printf("str2[] = %s\n", str2);
```

```
    }
    return 0;
}

void copystring_array(char str1[], char str2[]) {
    printf("String copying using array\n");
    int i;
    for (i = 0; str1[i] != '\0'; i++) {
        str2[i] = str1[i];
    }
    str2[i] = '\0';  // Null-terminate str2
}

void copystring_pointer(char *str1, char *str2) {
    printf("String copying using pointer\n");
    for (; *str1 != '\0'; ++str1, ++str2) {
        *str2 = *str1;  // Copy character from str1 to str2
    }
    *str2 = '\0';  // Null-terminate str2
}
```

ASSIGNMENT

1)
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[30];
    char s[30];
    char temp[30];
    int j = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);


    for (int i = 0; str[i] != '\0'; i++) {
        if (isalpha(str[i])) {
            s[j++] = tolower(str[i]);
        }
    }
    s[j] = '\0';


    strcpy(temp, s);


    int x = strlen(s);
    for (int i = 0; i < x / 2; i++) {
        char temp_char = s[i];
        s[i] = s[x - 1 - i];
        s[x - 1 - i] = temp_char;
    }


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


    if (strcmp(temp, s) == 0) {
```

```
        printf("Palindrome\n");
    } else {
        printf("Not a palindrome\n");
    }

    return 0;
}
```

2)

DYNAMIC MEMORY ALLOCATION

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int *ptr;
    int num,i;
    printf("enter no of elements:");
    scanf("%d",&num);

    printf("the number entered is n=%d\n",num);
    //dynamically allocatte memory for array
    ptr=(int *)malloc(num*sizeof(int));

    //check whether the memory is allocated successfully or not

    if(ptr==NULL){
        printf("memory not allocatewd\n");
        exit (0);
```

```
    }else{
        printf("memory is allocated\n");
    }

    for(i=0;i<num;i++){
        ptr[i]=i+1;
    }

    for(i=0;i<num;i++){
       printf("%d ,",ptr[i]);
    }

    free(ptr);



    return 0;
}
```

ASSIGNMENT

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';
f
 indLongestRepeatingSubstring(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");
    }}
================================================================
============================
```