1) WAP for copying strings

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

int main()
{
   char str1[10];
   char str2[10];
   strcpy(str1, "hello");
   strcpy(str2, "world");

   printf("str1[] = %s \t str2[] = %s",str1,str2);

   return 0;
}
```

2) WAP for understand the concept of strchr

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

int main()
{
   char str[] = "hi my name is vaisakh";
   int len = strlen(str);
   for(int i; i < len; i++){
      printf("str[%d] = %c ,address = %p\n ",i,str[i],(str+i));
   }
   char ch = 'm';
   char *pfound = NULL;
   pfound = strchr(str,ch);

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

   return 0;
}
```

o/p

```
 str[0] = h ,address = 0x7fff319fb7a0
 str[1] = i ,address = 0x7fff319fb7a1
 str[2] =   ,address = 0x7fff319fb7a2
 str[3] = m ,address = 0x7fff319fb7a3
 str[4] = y ,address = 0x7fff319fb7a4
```

```
str[5] =   ,address = 0x7fff319fb7a5
str[6] = n ,address = 0x7fff319fb7a6
str[7] = a ,address = 0x7fff319fb7a7
str[8] = m ,address = 0x7fff319fb7a8
str[9] = e ,address = 0x7fff319fb7a9
str[10] =   ,address = 0x7fff319fb7aa
str[11] = i ,address = 0x7fff319fb7ab
str[12] = s ,address = 0x7fff319fb7ac
str[13] =   ,address = 0x7fff319fb7ad
str[14] = v ,address = 0x7fff319fb7ae
str[15] = a ,address = 0x7fff319fb7af
str[16] = i ,address = 0x7fff319fb7b0
str[17] = s ,address = 0x7fff319fb7b1
str[18] = a ,address = 0x7fff319fb7b2
str[19] = k ,address = 0x7fff319fb7b3
str[20] = h ,address = 0x7fff319fb7b4
pfound = 0x7fff319fb7a3
```

3)WAP for extracting a word from a string

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

int main()
{
   char text[] = "every dog has his day";
   char word[] = "dog";

   char *pfound = NULL;
   pfound = strstr(text,word);

   if(pfound != NULL) {
      printf("The word \"%s\" is found at address: %p\n", word, (void *)pfound);
   } else {
      printf("The word \"%s\" is not found in the text.\n", word);
   }

   if (pfound != NULL) {
      printf("The word \"%s\" is found at address: %p\n", word, (void *)pfound);


   char extractedWord[sizeof(word)];
      strncpy(extractedWord, pfound, strlen(word));
      extractedWord[strlen(word)] = '\0';

      printf("Extracted word: %s\n", extractedWord);
   } else {
```

```
        printf("The word \"%s\" is not found in the text.\n", word);
    }


    return 0;
}
```

4) a) WAP for understand about strtok

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

int main()
{
   char str[80] = "hello my - name is - vaisakh";
   const char s[2] = "-";

   char *token = NULL;

   token = strtok(str, s);

   while(token != NULL){
      printf("token = %s \n",token);
      token = strtok(NULL, s);
   }

    return 0;
}
```

0/p

```
token = hello my
token =  name is
token =  vaisakh
```

b)//same prgm

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

int main()
{
```

```
   char str[80] = "hello my - name is - vaisakh";
   const char s[2] = " ";

   char *token = NULL;

   token = strtok(str, s);

   while(token != NULL){
      printf("token = %s \n",token);
      token = strtok(NULL, s);
   }

    return 0;
}
```

o/p

```
token = hello
token = my
token = -
token = name
token = is
token = -
token = vaisakh
```

5) WAP for some of string functions like find alphabets,digits and punctuations

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

int main()
{
  char buf[100];
  int nLetters = 0;
  int nDigits = 0;
  int nPunct = 0;

  printf("enter an interesting string of less than %d characters:\n",100);
  scanf("%s",buf);

  int i =0;
  while(buf[i])
  {
     if(isalpha(buf[i]))
```

```
      ++nLetters;

      else if(isdigit(buf[i]))
      ++nDigits;

      else if(ispunct(buf[i]))
      ++nPunct;

      ++i;
  }

   printf("\n your string contained %d letters, %d digits and %d punctuation characters
\n",nLetters,nDigits,nPunct);
     return 0;
}
```

6)WAP for Searching a string

```
#include <stdio.h>

int main()
{
 char text[100];
 char substring[40];

 printf("enter the string to be searched(less than %d characters):\n",100);
 scanf("%s", text);

 printf("enter the string sought(less than %d characters):\n",40);
 scanf("%s", substring);

 printf("\n first string entered:\n %s \n",text);
 printf("\n second string entered:\n %s \n",substring);

 // convert both strings to uppercase

 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;
}
```

7) 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>

#include<ctype.h>

int main() {

char s[100];

char r[100];

printf("Enter a String :");

scanf("%s",s);

int l=strlen(s);

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

r[i]=s[l-i-1];

}

r[l]='\0';

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

s[i] = tolower(s[i]);

r[i] = tolower(r[i]);

}

if (strcmp(r,s)!=0){

printf("\nNot Palindrome");
```

```
}

else{

printf("\n Palindrome");

}
}
```

8) 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

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

// Function to convert a string to lowercase
void toLowerCase(char *str) {
    for (int i = 0; str[i]; i++) {
        str[i] = tolower(str[i]); // Convert each character to lowercase
    }
}

int main() {
    char str[100];
    char words[50][50];
    int freq[50] = {0};
    int wordcount = 0;

    printf("Enter a string: ");
    // Using scanf to read a full line of input
    scanf(" %[^\n]s", str); // Reads until a newline is encountered

    toLowerCase(str); // Convert the input string to lowercase
```

```
    // Tokenize the string by spaces
    char *token = strtok(str, " ");
    while (token != NULL) {
        int found = 0;

        // Check if the word already exists in the array
        for (int i = 0; i < wordcount; i++) {
            if (strcmp(words[i], token) == 0) {
                freq[i]++;
                found = 1;
                break;
            }
        }

        // If the word is not found, add it to the array
        if (!found) {
            strcpy(words[wordcount], token);
            freq[wordcount] = 1;
            wordcount++;
        }

        // Get the next token
        token = strtok(NULL, " ");
    }

    // Print word frequencies
    printf("\nWord Frequencies:\n");
    for (int i = 0; i < wordcount; i++) {
        printf("Word: %s, Frequency: %d\n", words[i], freq[i]);
    }

    return 0;
}
```

9) 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 findAndReplace(char *str, const char *target, const char *replace) {
    char result[1000]; // Buffer to store the modified string
    int i = 0, j = 0;
    int targetLen = strlen(target);
    int replaceLen = strlen(replace);

    while (str[i] != '\0') {

        char *pos = strstr(&str[i], target);

        if (pos == &str[i]) {
            strcpy(&result[j], replace);
            j += replaceLen;
            i += targetLen;
        } else {

            result[j++] = str[i++];
        }
    }

    result[j] = '\0';

    strcpy(str, result);
}

int main() {
    char str[1000], target[100], replace[100];

    printf("Enter the string: ");
    scanf(" %[^\n]", str);

    printf("Enter the target substring: ");
    scanf(" %[^\n]", target);

    printf("Enter the replacement substring: ");
    scanf(" %[^\n]", replace);

    findAndReplace(str, target, replace);

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

    return 0;
}
```

o/p

```
Enter the string: helloworld
Enter the target substring: wor
Enter the replacement substring: ABCD
Modified string: helloABCDld
```

10) 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>

void reverseWords(const char *input, char *output) {
    char words[100][100]; // Array to store words
    int Count = 0;

    char temp[1000];
    strcpy(temp, input);


    char *token = strtok(temp, " ");
    while (token != NULL) {
        strcpy(words[Count], token);
        Count++;
        token = strtok(NULL, " ");
    }

    output[0] = '\0';
    for (int i = Count - 1; i >= 0; i--) {
        strcat(output, words[i]);
        if (i > 0) {
            strcat(output, " ");
        }
    }
}

int main() {
    char input[1000], output[1000];

    printf("Enter a sentence: ");
    scanf(" %[^\n]", input);
```

```
    reverseWords(input, output);

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

    return 0;
}
```

o/p

```
Enter a sentence: hi i am vaisakh
Reversed sentence: vaisakh am i hi
```

11) 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(const char *str, char *result) {
    int len = strlen(str);
    int maxLen = 0;

    for (int subLen = 1; subLen < len; subLen++) {

        for (int i = 0; i <= len - subLen; i++) {
            char substr[100] = {0};
            strncpy(substr, &str[i], subLen);

            for (int j = i + 1; j <= len - subLen; j++) {
                char compareStr[100] = {0};
                strncpy(compareStr, &str[j], subLen);

                if (strcmp(substr, compareStr) == 0) {

                    if (subLen > maxLen) {
                        maxLen = subLen;
                        strcpy(result, substr);
                    }
                }
```

```
            }
        }
    }
}

int main() {
    char str[1000], result[100] = {0};

    printf("Enter a string: ");
    scanf("%s", str);

    findLongestRepeatingSubstring(str, result);

    if (strlen(result) > 0) {
        printf("Longest repeating substring: %s\n", result);
    } else {
        printf("No repeating substring found.\n");
    }

    return 0;
}
```

12) WAP for using malloc()

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

int main()
{
    int *ptr;
    int num , i;

    printf("enter no 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 whether the memory is allocated successfully or not
    if(ptr == NULL){
        printf("memory not allocated");
        exit(0);
    }else{
```

```
        printf("memory 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;
}
```

o/p

```
enter no of elements 5

the number entered is n = 5
memory allocated successfully
1, 2, 3, 4, 5,
```