1.

```
#include<stdio.h>
#include<string.h>
int main(){
    char A[10]="Bhavana";
    char B[10]="Baiju";
    printf("strcmp(\"A\",\"A\") is ");
    printf("%d\n",strcmp("A","A"));


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

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

    printf("strcmp(\"A\",\"D\") is ");
    printf("%d\n",strcmp("A","D"));//Str1<str2

    printf("strcmp(\"Z\",\"D\") is ");
    printf("%d\n",strcmp("Z","D"));//Str1>Str2

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

    printf("strcmp(\"Apple\",\"Apple\") is ");
    printf("%d\n",strcmp("Apples","Apple"));

    printf("strcmp(\"ABBC\",\"AABCD\") is ");
    printf("%d\n",strcmp("ABBC","AABCD"));

    printf("strcmp(\"Astrounding\",\"Astro\") is ");
    printf("%d\n",strcmp("Astounding","Astro"));

    printf("strncmp(\"Astrounding\",\"Astro\") is ");
    printf("%d\n",strncmp("Astounding","Astso",5));//The difference btw the 
character is printed


    return 0;
}
```

2.

```
#include <stdio.h>
#include<string.h>
int main()
{
```

```c
    char str[] = "Hi my name is Bhavana";
    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);//For searching the character in the string

    printf("The address where the character %c is present in the string is 
%p",ch,pFound);






    return 0;
}
```

3.

```c
#include <stdio.h>
#include<string.h>
int main()
{
    char str[] = "Every dog has his day.";

    char word[] = "dog";


    char *pFound = NULL;


    pFound = strstr(str,word);//For searching the character in the string

    printf("The address where the character %s is present in the string is 
%p",word,pFound);
```

```
    return 0;
}
```

4.

```
#include <stdio.h>
#include<string.h>
int main()
{
    char buf[100] = "Every dog has his day.";

    int nLetters = 0;
    int nDigits = 0;
    int nPunct = 0;
    printf("\nEnter an interesting string of less than %d characters : ",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("\nYour string contained %d letters, %d digits , %d punctuations 
characters.\n",nLetters,nDigits,nPunct);






    return 0;
}
```

5.

```
#include <stdio.h>
#include<string.h>
int main()
{
    char text[100];
    char substring[40];


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

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


    printf("\nFirst string entered : %s ",text);
    printf("\nSecond string entered : %s ",substring);


    //convert both the string 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("\nThe Second string %s found in the first.\n
",((strstr(text,substring)==NULL)? "was not" : "was"));






    return 0;
}
```

6.

```
//Cocept string copy
#include<stdio.h>
#include<string.h>

void arr_copyString(char to[],char from[]);
void ptr_copyString(char *to,char *from);
int main()
{
```

```c
    char A[50];
    char B[20]="Anaswara";

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

    switch(op)
    {
        case 'a':
        arr_copyString(A,B);
        printf("\n The copied content is %s\n",A);
        break;
        case 'p':
        ptr_copyString(A,B);
        printf("\n The copied content is %s\n",A);
        break;
    }
}

void arr_copyString(char to[],char from[])
{
    int i;
    printf("Inside array copying\n");
    for(i=0;from[i] != '\0';i++)
    {
        to[i] = from[i];

    }
    to[i] = '\0';

}


void ptr_copyString(char *to, char *from)
{
    char *originalTo = to;
    printf("Inside pointer copying\n");
    for(;*from != '\0';from++,to++)
    {
        *to = *from;

    }
    *to = '\0';

}
```

7.

```c
/*
==============================================================================
====


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>

// Function to check if a string is a palindrome
int isPalindrome(char str[]) {
    int left = 0, right = strlen(str) - 1;

    while (left < right) {
        // Skip non-alphanumeric characters
        while (left < right && !isalnum(str[left])) {
            left++;
        }
        while (left < right && !isalnum(str[right])) {
            right--;
        }

        // Compare characters, ignoring case
        if (tolower(str[left]) != tolower(str[right])) {
            return 0; // Not a palindrome
        }
        left++;
        right--;
    }
    return 1; // Palindrome
}

int main() {
    char input[1000];

    printf("Enter a string: ");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = '\0'; // Remove trailing newline

    if (isPalindrome(input)) {
        printf("Palindrome\n");
    } else {
        printf("Not a Palindrome\n");
    }

    return 0;
}
```

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]);
    }
}

// Function to count word frequencies
void countWordFrequencies(char str[]) {
    char tempStr[1000];
    strcpy(tempStr, str); // Make a copy of the input string to tokenize
```

```c
    char *words[500]; // Array to store unique words
    int frequencies[500] = {0}; // Array to store frequencies
    int wordCount = 0;

    char *token = strtok(tempStr, " .,!?;:\n"); // Tokenize the string
    while (token != NULL) {
        toLowerCase(token); // Convert token to lowercase

        int found = 0;
        for (int i = 0; i < wordCount; i++) {
            if (strcmp(words[i], token) == 0) { // Word already exists
                frequencies[i]++;
                found = 1;
                break;
            }
        }

        if (!found) { // New word found
            words[wordCount] = token;
            frequencies[wordCount] = 1;
            wordCount++;
        }

        token = strtok(NULL, " .,!?;:\n");
    }

    // Print the words and their frequencies
    printf("Word Frequencies:\n");
    for (int i = 0; i < wordCount; i++) {
        printf("Word: %s, Frequency: %d\n", words[i], frequencies[i]);
    }
}

int main() {
    char input[1000];

    printf("Enter a string: ");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = '\0'; // Remove trailing newline

    countWordFrequencies(input);

    return 0;
}
```

9.

```c
/*
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>

// Function to replace all occurrences of a target substring with another
substring
void findAndReplace(char str[], char target[], char replacement[]) {
    char result[1000]; // Resulting string after replacements
    int i = 0, j = 0;
    int targetLen = strlen(target);
    int replacementLen = strlen(replacement);

    while (str[i] != '\0') {
        // Check if the target substring matches
        if (strstr(&str[i], target) == &str[i]) {
            // Copy the replacement string to the result
            strcpy(&result[j], replacement);
            j += replacementLen;
            i += targetLen; // Skip over the target substring
        } else {
            // Copy the current character
            result[j++] = str[i++];
        }
    }

    result[j] = '\0'; // Null-terminate the result string
    strcpy(str, result); // Copy the result back to the original string
}

int main() {
    char str[1000], target[100], replacement[100];
    printf("Enter the string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0'; // Remove trailing newline

    printf("Enter the target substring: ");
    fgets(target, sizeof(target), stdin);
    target[strcspn(target, "\n")] = '\0'; // Remove trailing newline

    printf("Enter the replacement substring: ");
    fgets(replacement, sizeof(replacement), stdin);
    replacement[strcspn(replacement, "\n")] = '\0'; // Remove trailing newline

    findAndReplace(str, target, replacement);

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

    return 0;
}
```

10.

```c
/*

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>

// Function to reverse words in a sentence
void reverseWords(char str[]) {
    char *words[100]; // Array to store individual words
    int wordCount = 0;

    // Tokenize the sentence into words
    char *token = strtok(str, " ");
    while (token != NULL) {
        words[wordCount++] = token; // Store the word
        token = strtok(NULL, " ");
    }

    // Rebuild the sentence with words in reverse order
    char reversed[1000] = "";
    for (int i = wordCount - 1; i >= 0; i--) {
        strcat(reversed, words[i]);
        if (i > 0) { // Add a space between words, but not after the last word
            strcat(reversed, " ");
        }
    }

    printf("Reversed Sentence: %s\n", reversed);
}

int main() {
    char input[1000];

    printf("Enter a sentence: ");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = '\0'; // Remove trailing newline

    reverseWords(input);

    return 0;
}
```

11.

```c
/*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>

// Function to find the longest repeating substring
void longestRepeatingSubstring(char str[]) {
    int len = strlen(str);
    char longest[100] = "";
    int maxLen = 0;
    // Iterate through all possible substrings
    for (int i = 0; i < len; i++) {
        for (int j = i + 1; j < len; j++) {
            int k = 0;

            // Compare characters in substrings
            while (i + k < len && j + k < len && str[i + k] == str[j + k]) {
                k++;
            }

            // Update the longest repeating substring
            if (k > maxLen) {
                maxLen = k;
                strncpy(longest, &str[i], k);
                longest[k] = '\0'; // Null-terminate the substring
            }
        }
    }

    if (maxLen > 0) {
        printf("Longest Repeating Substring: %s\n", longest);
    } else {
        printf("No Repeating Substring Found.\n");
    }
}

int main() {
    char input[1000];

    printf("Enter a string: ");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = '\0'; // Remove trailing newline

    longestRepeatingSubstring(input);

    return 0;
}
```

12.

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

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

    //Dynamically allocated memory for the array
    ptr = ((int *) malloc(num + sizeof(int)));

    //Check whether the memory is allocated successfully or not

    if(ptr == NULL){
        printf("\nMemory not allocated\n");
        exit(0);

    }else{
        printf("\nMemory is allocated successfully\n");
    }

    //For population or generating the array
    for(i=0;i<num;i++){
        ptr[i] = i+1;
    }


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


    //This is a standard practice to free the Dynamically allocated memory
    free(ptr);

    return 0 ;
}
```