Program to count the alphabets, digits and punctuations in a string

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
char buff[100];
int letter=0;
int digit=0;
int punc=0;
printf("enter a string \n");
gets(buff);
int i=0;
while(buff[i])
{
    if(isalpha(buff[i]))
    {
        letter++;
    }
    else if(isdigit(buff[i]))
    {
        digit++;
    }
    else if(ispunct(buff[i]))
    {
        punc++;
    }
    i++;
}
printf("the string contains %d letters %d digits and %d
punctuations\n",letter,digit,punc);
}
```

Program to find a particular character in a string

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char str1[]="the quick brown fox";
    for(int i=0;str1[i]!='\0';i++)
    {
        printf("address of %c is %p \n",str1[i],str1+i);
    }
```

```
    char ch='q';
    char *ptr=NULL;
    ptr=strchr(str1,ch);
    printf("%p",ptr);
}
```

Program to find a particular character in a string

```
#include<stdio.h>
#include<string.h>
int main()
{
    char str1[]="the quick brown fox";
    for(int i=0;str1[i]!='\0';i++)
    {
        printf("address of %c is %p \n",str1[i],str1+i);
    }
    char word[]="quick";
    char *ptr=NULL;
    ptr=strstr(str1,word);
        for(int i=0;*ptr!=' ';i++)
       printf("%s\n",ptr);
    printf("%p",ptr);
}
```

Program to demonstrate pointer and string

```
#include<stdio.h>
void Array(char arr3[],char arr4[]);
void ptr(char *arr5, char *arr6);
int main()
{
    char n;
    char arr1[20]="vivek";
    char arr2[20];
    printf("enter a or p \n");
    scanf("%c",&n);
    switch(n)
    {
        case 'a':
            Array(arr1,arr2);
```

```
            break;
        case 'p':
            ptr(arr1,arr2);
            break;
        default:
        printf("wrong choice \n");
    }
}
void Array(char arr3[],char arr4[])
{
    int i;
    for(i=0;arr3[i]!='\0';i++)
    {
        arr4[i]=arr3[i];
    }
    arr4[i]='\0';
    printf("string is %s",arr4);
}
void ptr(char *arr5, char *arr6)
{
    char *p = arr6;
    while (*arr5 != '\0')
    {
        *arr6 = *arr5;
        arr5++;
        arr6++;
    }
    *arr6 = '\0';
    printf("String is: %s\n", p);
}
```

Program to convert lowercase string to uppercase

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
    char text[100];
    char substring[40];
    printf("enter the string to be searched \n");
    scanf("%s",text);
    printf("enter the string to be found \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"));
}
```

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>
void remov(const char *str1, char *rem);
int palindrome(const char *str);
int main()
{
    char str[50];
    char rem[50];
    printf("Enter a string: ");
    gets(str);
    str[strcspn(str, "\n")] = '\0';
    remov(str, rem);
    if (palindrome(rem))
    {
        printf("Palindrome\n");
    } else
    {
        printf("Not a palindrome\n");
    }
    return 0;
}
void remov(const char *str1, char *rem)
{
    int j = 0;
```

```
    for (int i = 0; str1[i] != '\0'; i++)
    {
        if (isalnum(str1[i]))
        {
            rem[j++] = tolower(str1[i]);
        }
    }
    rem[j] = '\0';
}
int palindrome(const char *str)
{
    int left = 0;
    int right = strlen(str) - 1;

    while (left < right)
    {
        if (str[left] != str[right])
        {
            return 0;
        }
        left++;
        right--;
    }
    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

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void lower(char *str);
```

```c
int main()
{
    char n[1000];
    char *words[100];
    int freq[100] = {0};
    int count = 0;
    printf("Enter a string: ");
    gets(n);
    n[strcspn(n, "\n")] = '\0';
    lower(n);
    char *token = strtok(n, " .,!?;");
    while (token != NULL)
    {
        int found = 0;
        for (int i = 0; i<count; i++)
        {
            if (strcmp(words[i], token) == 0)
            {
                freq[i]++;
                found = 1;
                break;
            }
        }
        if (!found)
        {
            words[count] = token;
            freq[count] = 1;
            count++;
        }
        token = strtok(NULL, " .,!?;");
    }
    printf("\nWord Frequencies:\n");
    for (int i = 0; i < count; i++) {
        printf("Word: %s, Frequency: %d\n", words[i], freq[i]);
    }
    return 0;
}
void lower(char *str)
{
    for (int i = 0; str[i]; i++)
    {
        str[i] = tolower(str[i]);
    }
}
```

==========

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 buffer[1000];
    char *pos;
    int targetLen = strlen(target);
    int replaceLen = strlen(replace);
    int index = 0;
    buffer[0] = '\0';
    while ((pos = strstr(str, target)) != NULL)
    {
        int lenBefore = pos - str;
        strncpy(buffer + index, str, lenBefore);
        index += lenBefore;
        strcpy(buffer + index, replace);
        index += replaceLen;
        str = pos + targetLen;
    }
    strcpy(buffer + index, str);
    strcpy(str, buffer);
}
int main() {
    char str[1000];
    char target[100], replace[100];
    printf("Enter the original string: ");
    gets(str);
```

```
    str[strcspn(str, "\n")] = '\0';
    printf("Enter the target substring: ");
    gets(target);
    target[strcspn(target, "\n")] = '\0';
    printf("Enter the replacement substring: ");
    gets(replace);
    replace[strcspn(replace, "\n")] = '\0';
    findAndReplace(str, target, replace);
    printf("Modified string: %s\n", str);
    return 0;
}
```

==========

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 reverse(char *sentence);
int main()
{
    char sentence[100];
    printf("Enter a sentence: ");
    gets(sentence);
    sentence[strcspn(sentence, "\n")] = '\0';
    reverse(sentence);
    return 0;
}
void reverse(char *sentence)
{
    char *words[100];
    int count = 0;
    char *token = strtok(sentence, " ");
    while (token != NULL)
```

```
    {
        words[count++] = token;
        token = strtok(NULL, " ");
    }
    char reversed[100] = "";
    for (int i = count - 1; i >= 0; i--)
    {
        strcat(reversed, words[i]);
        if (i > 0)
        {
            strcat(reversed, " ");
        }
    }
    printf("Reversed sentence: %s\n", reversed);
}
```

==========

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 substring(char *str);
int main()
{
    char str[1000];
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    substring(str);
    return 0;
}
void substring(char *str)
{
    int len = strlen(str);
    char longest[100] = "";
    int max = 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 > max)
            {
                max= k;
                strncpy(longest, str + i, k);
                longest[k] = '\0';
            }
        }
    }
    if (max> 0)
    {
        printf("Longest repeating substring: %s\n", longest);
    } else
    {
        printf("No repeating substring found.\n");
    }
}
```

Dynamic Memory Allocation

```
#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);
    ptr = (int *)malloc(num * sizeof(int));
    if(ptr == NULL)
    {
        printf("Memory not allocated \n");
        exit(0);
```

```
    }else{
        printf("Memory is allocated successfully \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;
}
```