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 str[50];

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

    scanf("%[^\n]",str);


    int length=strlen(str);

    int i,j=length-1;


    for(i=0;i<j;){

        if(!isalpha(str[i])){

            i++;

            continue;

        }

        if(!isalpha(str[j])){

            j--;

            continue;

        }

        if(tolower(str[i])!=tolower(str[j])){
            printf("not pallindrome\n");
            return 0;
        }
        i++;
        j--;

    }printf("pallindrome");
}
```

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>
int main()
{
    char *word[10] = {NULL};
    int count[10] = {0};
```

```
char str[50];
char temp[50];
printf("Input: ");
scanf(" %[^\n]", str);

strcpy(temp, str);

int i = 0, found = 0;
char *token = strtok(temp, " .,!?");
while (token != NULL)
{
    found = 0;
    for (int j = 0; j < i; j++)
    {
        if (strcmp(word[j], token) == 0)
        {
            count[j]++;
            found = 1;
            break;
        }
    }

    if (!found)
    {
        word[i] = token;
        count[i]++;
        i++;
    }

    token = strtok(NULL, " .,!?");
}
```

```
    for (int j = 0; j < i; j++)
    {
        printf("Word:%s, Frequency: %d\n", word[j], count[j]);
    }

    return 0;
}
```

__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 replaceSubstring(char *, const char *, const char *);

int main()
{
    char str[100], target[50], replace[50];

    printf("String: ");
    scanf(" %[^\n]", str);
    printf("Target: ");
    scanf(" %[^\n]", target);


    printf("Replace with: ");
    scanf(" %[^\n]", replace);
    replaceSubstring(str, target, replace);
    printf("Modified string: %s\n", str);
    return 0;
}


void replaceSubstring(char *str, const char *target, const char *replace)
{
    char buffer[100];
    char *pos;
    int targetLen = strlen(target);
    int replaceLen = strlen(replace);
    char *current = str;
    while ((pos = strstr(current, target)) != NULL)
    {
        strncat(buffer, current, pos - current);


        strcat(buffer, replace);


        current = pos + targetLen;
    }
    strcat(buffer, current);
    strcpy(str, buffer);
}
```

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 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);
}
```

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 findLongest(char *);

int main()
{
    char str[100];
    printf("Input: ");
    scanf("%s", str);

    findLongest(str);

    return 0;
}
```

```
void findLongest(char *str)
{
    int n = strlen(str);
    int maxLength = 0;
    char longestSub[100];
    for (int len = 1; len < n; len++)
    {
        for (int i = 0; i <= n - len; i++)
        {
            for (int j = i + 1; j <= n - len; j++)
            {
                if (strncmp(str + i, str + j, len) == 0)
                {
                    if (len > maxLength)
                    {
                        maxLength = len;
                        strncpy(longestSub, str + i, len);
                        longestSub[len] = '\0';
                    }
                    break;
                }
            }
        }
    }
    if (maxLength > 0)
    {
        printf("Longest repeated substring: \"%s\"\n", longestSub);
    }
    else
    {
        printf("No repeated substring found.\n");
    }
}
```