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 input[100];

   int i=0,j=0;

   char temp[100];

   printf("Enter input: ");

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

   while (input[i] != '\0')

   {

      if (input[i] != ' ' && input[i] != ',')

      {

         input[j++] = input[i];

      }

      i++;

   }

   input[j] = '\0';
```

```
    for(int i = 0; i<strlen(input); i++)
    {
        input[i] = tolower(input[i]);
    }



    j=0;
    for(int i=strlen(input)-1;i>=0;i--)
    {
        temp[j]=input[i];
        j++;
    }

    if(strcmp(input,temp)==0)
    {
        printf("Palindrome");

    }
    else
    {
        printf("not Palindrome");
    }



    return 0;
}
```

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
```

```c
#include <stdio.h>

#include <string.h>

int main()

{

 char *word[10] = {NULL};

 int frequency[10] = {0};

 char input[50];

 char temp[50];


 printf("Input: ");

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


 strcpy(temp, input);
```

```
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)

    {

    frequency[j]++;

    found = 1;

    break;

  }

}



if (!found)

{

  word[i] = token;

  frequency[i]++;

  i++;

}


token = strtok(NULL, " .");

}


for (int j = 0; j < i; j++)
```

```
 {
 printf("Word: %s , Frequency: %d\n", word[j], frequency[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 findandreplace(char *input,char *target, char *replace);
int main()
{
   char input[100], target[100], replace[100];
   printf("Enter the string: ");
   scanf(" %[^\n]",input);
   printf("Enter the target: ");
   scanf("%s", target);
   printf("Enter the replace: ");
```

```
    scanf("%s", replace);
    findandreplace(input, target, replace);


    return 0;
}

void findandreplace(char *input, char *target,char *replace)
{
    char result[100] = "";
    char *pos;
    int targetlen = strlen(target);
    int replacelen = strlen(replace);
    while ((pos = strstr(input, target)) != NULL)
    {
        strncat(result, input, pos - input);
        strcat(result, replace);
        input = pos + targetlen;
    }
    strcat(result, input);
    printf("Modified string: %s\n", result);
}
```

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 rev(char *);

int main()

{

 char str[50];

 printf("Input: ");

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


 rev(str);

 char *token = strtok(str, " ");

 char buffer[100]="";

 while (token != NULL)

 {

 rev(token);

 strcat(buffer, token);

 strcat(buffer, " ");

 token = strtok(NULL, " ");

 }

 printf("%s", buffer);

 return 0;

}

void rev(char str[])

{

 int i = 0;

 int j = strlen(str) - 1;

 while (i < j)

 {
```

```
 char temp = str[i];

 str[i] = str[j];

 str[j] = temp;

 i++;

 j--;

 }

}
```

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 *str);


int main()

{

    char str[100];

    printf("Input: ");

    scanf("%s", str);


    findlongest(str);


    return 0;
```

```
}




void findlongest(char *str)
{
 int n = strlen(str);
 int maxlen = 0;
 char longsub[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 > maxlen)
             {
                maxlen = len;
                strncpy(longsub, str + i, len);
                longsub[len] = '\0';
             }
          break;
          }
       }
    }
 }
 if (maxlen > 0)
 {
    printf("Longest repeated substring is %s ", longsub);
 }
else
{
    printf("No repeated substring.");
}
}
```