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 isPalindrome(char str[]) {
    int left = 0, right = strlen(str) - 1;

    while (left < right) {
        while (left < right && !isalnum(str[left])) left++;
        while (left < right && !isalnum(str[right])) right--;
        if (tolower(str[left]) != tolower(str[right])) return 0;
        left++;
        right--;
    }
    return 1;
}

int main() {
    char str[1000];
    printf("Enter a string: ");
    scanf("%[^\n]", str);

    if (isPalindrome(str))
```

```
        printf("Palindrome\n");
    else
        printf("Not a palindrome\n");

    return 0;
}
```

OUTPUT

Enter a string: Hello, World!

Not a palindrome

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.

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

void toLowerCase(char* str) {
    for (int i = 0; str[i]; i++) {
        str[i] = tolower(str[i]);
    }
}

int main() {
    char str[1000], words[100][50];
    int freq[100] = {0}, count = 0;

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

```
    toLowerCase(str);

    char* token = strtok(str, " .,!?;:-");
    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) {
            strcpy(words[count], token);
            freq[count]++;
            count++;
        }
        token = strtok(NULL, " .,!?;:-");
    }

    printf("\nWord Frequencies:\n");
    for (int i = 0; i < count; i++) {
        printf("%s: %d\n", words[i], freq[i]);
    }

    return 0;
}
```

OUTPUT

Enter a string: Hello world Hello everyone. Welcome to the world.

Word Frequencies:

hello: 2

world: 2

everyone: 1

welcome: 1

to: 1

the: 1

3. 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.

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

void replaceSubstring(char* str, const char* target, const char* replacement) {
   char buffer[1000];
   char* pos;
   int targetLen = strlen(target);
   int replacementLen = strlen(replacement);

   buffer[0] = '\0';

   while ((pos = strstr(str, target)) != NULL) {
      strncat(buffer, str, pos - str);
      strcat(buffer, replacement);
      str = pos + targetLen;
```

```
    }
    strcat(buffer, str);
    strcpy(str, buffer);
}

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

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

    replaceSubstring(str, target, replacement);

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

    return 0;
}
```

OUTPUT

Hello world. Hello everyone.

Hello

Hi

Hi world. Hi everyone.

4. 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.

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

```
#include <string.h>

int main() {
   char str[1000], reversed[1000] = "";
   char* token;

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

   token = strtok(str, " ");
   while (token != NULL) {
      char temp[1000];
      strcpy(temp, token);
      strcat(temp, " ");
      strcat(temp, reversed);
      strcpy(reversed, temp);
      token = strtok(NULL, " ");
   }

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

   return 0;
}
```

OUTPUT

```
Enter a sentence: Hello world
Reversed sentence: world Hello
```

5. 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.

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

void findLongestSubstring(char* str) {
  int len = strlen(str);
  int maxLength = 0;
  char longestSubstr[1000] = "";

  for (int i = 0; i < len; i++) {
    for (int j = i + 1; j < len; j++) {
      int subLen = j - i + 1;
      char substr[1000];
      strncpy(substr, &str[i], subLen);
      substr[subLen] = '\0';

      for (int k = 0; k < len; k++) {
        if (k != i && strncmp(&str[k], substr, subLen) == 0) {
          if (subLen > maxLength) {
            maxLength = subLen;
            strcpy(longestSubstr, substr);
          }
          break;
        }
      }
    }
  }
```

```
    if (maxLength > 0) {
        printf("Longest substring that appears more than once: %s\n", longestSubstr);
    } else {
        printf("No repeated substring found.\n");
    }
}

int main() {
    char str[1000];

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

    findLongestSubstring(str);

    return 0;
}
```

OUTPUT

```
Enter a string: abcaabcdabc
Longest substring that appears more than once: abc
```