12. **Write a program in C to count occurrences of a word in a string.**

Sample Input	Sample Output
"I liked the story about the sad giant" "the"	2
"It is what it is" "it"	1

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

char str[100], sub[100];
int count1 = 0;

void main()
{
    int i, j, l1, l2;
    printf("\n enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    printf("\n enter a substring: ");
    fgets(sub, sizeof(sub), stdin);
    sub[strcspn(sub, "\n")] = '\0';

    l1 = strlen(str);
    l2 = strlen(sub);

    for (i = 0; i <= l1 - l2; i++)
    {
        for (j = 0; j < l2; j++)
        {
            if (str[i + j] != sub[j])
                break;
        }
        if (j == l2)
        {
            count1++;
        }
    }

    printf("%s occurs %d times in %s\n", sub, count1, str);
}
```

15. **Write a program in C to reverse the words in a string.**

Sample Input	Sample Output
"My name is Andy"	"Andy is name My"
"Abc 123 7&*&*"	"7&*&* 123 Abc"

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

void reverseWords(char *str) {
    char tempStr[1000];
    char *words[100];
    int wordCount = 0;

    // Copy the original string to a temporary string
    strcpy(tempStr, str);

    // Tokenize the string into words using space as a delimiter
    char *token = strtok(tempStr, " ");
    while (token != NULL) {
        words[wordCount++] = token;
        token = strtok(NULL, " ");
    }

    // Print the words in reverse order
    for (int i = wordCount - 1; i >= 0; i--) {
        printf("%s", words[i]);
        if (i > 0) {
            printf(" ");
        }
    }
    printf("\n");
}

int main() {
    char string[1000];

    // Input the string from the user
    printf("Enter the string: ");
    fgets(string, sizeof(string), stdin);
    string[strcspn(string, "\n")] = 0; // Remove trailing newline character

    // Reverse the words in the string
    printf("Reversed string: ");
```

```
    reverseWords(string);

    return 0;
}
```

14. **Write a program in C to find the maximum occurring character in a string.**

Sample Input	Sample Output
"Welcome to CSE"	E (or e)
"mmmttssarrrddd"	D (or d)
"mmmttssarrrDDd"	D (or d)

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

char findMaxOccurringChar(const char *str) {
    int frequency[256] = {0}; // Array to store frequency of each character
    int maxFreq = 0;
    char maxChar = '\0';

    // Count the frequency of each character in the string
    for (int i = 0; str[i] != '\0'; i++) {
        char ch = tolower(str[i]); // Convert to lowercase for case insensitivity
        if (isalpha(ch)) { // Consider only alphabetic characters
            frequency[(int)ch]++;
            if (frequency[(int)ch] > maxFreq) {
                maxFreq = frequency[(int)ch];
                maxChar = ch;
            }
        }
    }

    return maxChar;
}

int main() {
    char string[1000];
```

```
    // Input the string from the user
    printf("Enter the string: ");
    fgets(string, sizeof(string), stdin);
    string[strcspn(string, "\n")] = 0; // Remove trailing newline character

    // Find the maximum occurring character
    char maxChar = findMaxOccurringChar(string);

    // Output the result
    if (maxChar != '\0') {
        printf("The maximum occurring character is: %c\n", toupper(maxChar));
    } else {
        printf("No valid alphabetic characters found in the string.\n");
    }

    return 0;
}
```

10. **Write a program in C to check whether a string is a palindrome or not.**

Sample Input	Sample Output
"My name is andy"	no
"madam"	yes

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

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

    // Compare characters from both ends of the string
    while (start < end) {
        // Compare characters in a case-insensitive manner
        if (tolower(str[start]) != tolower(str[end])) {
            return 0; // Not a palindrome
        }
        start++;
        end--;
```

```
    }
    return 1; // Palindrome
}

int main() {
    char str[100];

    // Input the string
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Remove the trailing newline character from fgets
    str[strcspn(str, "\n")] = 0;

    // Check if the string is a palindrome
    if (isPalindrome(str)) {
        printf("yes\n");
    } else {
        printf("no\n");
    }

    return 0;
}
```