```c
    }
}

// Function to print the deck of cards
void print_deck(Card deck[]) {
    for (int i = 0; i < DECK_SIZE; i++) {
        printf("%s of %s\n", deck[i].rank, deck[i].suit);
    }
}

int main() {
    srand(time(NULL)); // Seed the random number generator

    Card deck[DECK_SIZE];
    create_deck(deck); // Create the deck

    printf("Initial Deck:\n");
    print_deck(deck);  // Print the initial deck

    shuffle_deck(deck); // Shuffle the deck

    printf("\nShuffled Deck:\n");
    print_deck(deck);   // Print the shuffled deck

    return 0;
}
```

29. Display Fibonacci Sequence Using Recursion

```c
#include <stdio.h>

// Recursive function to calculate Fibonacci number
unsigned int fibonacci(unsigned int n) {
    if (n <= 1) return n; // Base case
    return fibonacci(n - 1) + fibonacci(n - 2); //
Recursive case
}

int main() {
    unsigned int num_terms;

    // Prompt user for the number of terms
    printf("Enter the number of terms in the Fibonacci
sequence: ");
    if (scanf("%u", &num_terms) != 1 || num_terms < 0) {
        printf("Please enter a valid non-negative
integer.\n");
        return 1;
    }

    // Display the Fibonacci sequence
    printf("Fibonacci sequence of %u terms:\n", num_terms);
    for (unsigned int i = 0; i < num_terms; i++) {
        printf("%u\n", fibonacci(i));
    }

    return 0;
}
```

30. Find Factorial of Number Using Recursion

```c
#include <stdio.h>

// Recursive function to calculate factorial
unsigned long long factorial(int n) {
    if (n == 0 || n == 1) return 1; // Base case
    return n * factorial(n - 1); // Recursive case
}

int main() {
    int number;

    // Prompt user for a non-negative integer
    printf("Enter a non-negative integer: ");
    if (scanf("%d", &number) != 1 || number < 0) {
        printf("Please enter a valid non-negative 
integer.\n");
        return 1;
    }

    // Calculate and display the factorial
    printf("The factorial of %d is: %llu\n", number, 
factorial(number));

    return 0;
}
```

31. Convert Decimal to Binary

```
#include <stdio.h>

// Function to convert decimal to binary
void decimal_to_binary(unsigned int num) {
    if (num == 0) {
        printf("0");
        return;
    }

    unsigned int binary[32]; // Array to store binary
digits
    int index = 0;

    // Convert decimal to binary
    while (num > 0) {
        binary[index++] = num % 2; // Store remainder
(binary digit)
        num /= 2; // Divide by 2
    }

    // Print binary in reverse order
    printf("The binary equivalent is: ");
    for (int i = index - 1; i >= 0; i--) {
        printf("%u", binary[i]);
    }
    printf("\n");
}

int main() {
    unsigned int decimal_number;

    // Prompt user for a decimal number
    printf("Enter a decimal number: ");
    if (scanf("%u", &decimal_number) != 1) {
        printf("Please enter a valid non-negative
integer.\n");
        return 1;
    }
    // Call the function to convert and display the binary
equivalent
    decimal_to_binary(decimal_number);

    return 0;
}
```

32. Find ASCII Value of Character

```c
#include <stdio.h>

int main() {
    char character;

    // Prompt user for a character
    printf("Enter a character: ");
    scanf(" %c", &character);  // Space before %c to skip
any whitespace

    // Print the ASCII value of the character
    printf("The ASCII value of '%c' is: %d\n", character,
(int)character);

    return 0;
}

/*
Enter a character: a
The ASCII value of 'a' is: 97

Enter a character: Z
The ASCII value of 'Z' is: 90
*/
```

33. Check Whether a String is Palindrome or Not

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

#define MAX_LENGTH 100

// Function to check if a string is a palindrome
int is_palindrome(const char *str) {
    int left = 0;
    int right = strlen(str) - 1;

    while (left < right) {
        if (str[left] != str[right]) {
            return 0; // Not a palindrome
        }
        left++;
        right--;
    }
    return 1; // Is a palindrome
}

// Function to clean the input string
void clean_string(const char *input, char *cleaned) {
    int j = 0;
    for (int i = 0; input[i] != '\0'; i++) {
        if (isalnum(input[i])) {
            cleaned[j++] = tolower(input[i]);
        }
    }
    cleaned[j] = '\0'; // Null-terminate the cleaned string
}

int main() {
    char input[MAX_LENGTH];
    char cleaned[MAX_LENGTH];
    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input, sizeof(input), stdin);

    // Clean the string
    clean_string(input, cleaned);

    // Check if the cleaned string is a palindrome
    if (is_palindrome(cleaned)) {
        printf("\"%s\" is a palindrome.\n", input);
    } else {
        printf("\"%s\" is not a palindrome.\n", input);
    }

    return 0;
}

/*

Enter a string: A man, a plan, a canal, Panama
"A man, a plan, a canal, Panama
" is a palindrome.

Enter a string: asasd
"asasd
" is not a palindrome.
*/
```

34. Sort Words in Alphabetical Order

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

#define MAX_WORDS 100
#define MAX_WORD_LENGTH 50

// Function to compare two strings for qsort
int compare(const void *a, const void *b) {
    return strcmp(*(const char **)a, *(const char **)b);
}

int main() {
    char input[500];  // Buffer to store the input sentence
    char *words[MAX_WORDS];  // Array to store pointers to
words
    int count = 0;  // Count of words

    // Prompt user for a sentence or a list of words
    printf("Enter a sentence or a list of words: ");
    fgets(input, sizeof(input), stdin);

    // Split the input into words
    char *token = strtok(input, " \n");
    while (token != NULL && count < MAX_WORDS) {
        words[count++] = token;  // Store the pointer to
the word
        token = strtok(NULL, " \n");
    }

    // Sort the words in alphabetical order
    qsort(words, count, sizeof(char *), compare);

    // Display the sorted words
    printf("Sorted Words:\n");
    for (int i = 0; i < count; i++) {
        printf("%s", words[i]);
        if (i < count - 1) {
```

```c
            printf(", ");
        }
    }
    printf("\n");

    return 0;
}

/*
Example Output:
Enter a sentence or a list of words: ad d dwqdwq qw qwd
sadasd wqdwq
Sorted Words:
ad, d, dwqdwq, qw, qwd, sadasd, wqdwq
*/
```

35. Replace Characters of a String

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

void replace_characters(char *str, char target, char
replacement) {
    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == target) {
            str[i] = replacement;
        }
    }
}

int main() {
    char input_string[100];
    char target_char, replacement_char;

    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input_string, sizeof(input_string), stdin);
    input_string[strcspn(input_string, "\n")] = 0; //
Remove newline character

    if (strlen(input_string) == 0) {
        printf("Please enter a valid string.\n");
        return 1;
    }

    // Prompt user for target and replacement characters
    printf("Enter the target character: ");
    scanf(" %c", &target_char); // Leading space to consume
any newline

    printf("Enter the replacement character: ");
    scanf(" %c", &replacement_char);

    // Replace characters in the input string
    replace_characters(input_string, target_char,
replacement_char);
```

```c
    // Display the modified string
    printf("Modified String: %s\n", input_string);

    return 0;
}

/*
Enter a string: Hello World!
Enter the target character: o
Enter the replacement character: O
Modified String: HellO WOrld!
*/
```

36. Reverse a String

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

void reverse_string(char *str) {
    int length = strlen(str);
    char temp;

    // Reverse the string in place
    for (int i = 0; i < length / 2; i++) {
        temp = str[i];
        str[i] = str[length - i - 1];
        str[length - i - 1] = temp;
    }
}

int main() {
    char input_string[100];

    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input_string, sizeof(input_string), stdin);
    input_string[strcspn(input_string, "\n")] = 0; //
Remove newline character

    // Check if input is a valid string
    if (strlen(input_string) > 0) {
        // Reverse the string
        reverse_string(input_string);

        // Display the reversed string
        printf("Reversed String: %s\n", input_string);
    } else {
        printf("Please enter a valid string.\n");
    }

    return 0;
}
```

```
/*
Enter a string: Hello World!
Reversed String: !dlroW olleH
*/
```

37. Check the Number of Occurrences of a Character in the String

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

int main() {
    char input_string[100];  // Array to hold the input
string
    char target_char;        // Variable to hold the
character to count
    int count = 0;          // Counter for occurrences

    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input_string, sizeof(input_string), stdin);
    input_string[strcspn(input_string, "\n")] = 0; //
Remove trailing newline

    // Prompt user for a character to count
    printf("Enter the character to count: ");
    scanf(" %c", &target_char); // Read a single character

    // Check if the input string is valid
    if (strlen(input_string) > 0) {
        // Count occurrences of the target character
        for (int i = 0; i < strlen(input_string); i++) {
            if (input_string[i] == target_char) {
                count++;
            }
        }

        // Display the result
        printf("Number of occurrences of '%c' in '%s':
%d\n", target_char, input_string, count);
    } else {
        printf("Please enter a valid string.\n");
    }
    return 0;
}

/*

Enter a string: Hello World!
Enter the character to count: l
Number of occurrences of 'l' in 'Hello World!': 3
*/
```

38. Convert the First Letter of a String into UpperCase

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

void capitalize_first_letter(char *str) {
    // Capitalize the first letter if the string is not
empty
    if (str[0] != '\0') {
        str[0] = toupper(str[0]);
    }
}

int main() {
    char input_string[100]; // Declare a string buffer with
a maximum length

    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input_string, sizeof(input_string), stdin); //
Read a line of input
    input_string[strcspn(input_string, "\n")] = 0; //
Remove trailing newline

    // Check if input is a valid string
    if (strlen(input_string) > 0) {
        // Capitalize the first letter
        capitalize_first_letter(input_string);

        // Display the result
        printf("Original String: %s\n", input_string);
        printf("String with First Letter Uppercase: %s\n",
input_string);
    } else {
        printf("Please enter a valid string.\n");
    }
    return 0;
}

/*
Enter a string: asd
Original String: Asd
String with First Letter Uppercase: Asd
*/
```

39. Count the Number of Vowels in a String

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

// Function to count vowels in a string
int count_vowels(const char *str) {
    int count = 0;
    const char *vowels = "aeiouAEIOU"; // Vowels to check
against

    // Iterate through each character in the string
    while (*str) {
        // Check if the character is a vowel
        if (strchr(vowels, *str)) {
            count++;
        }
        str++;
    }

    return count;
}

int main() {
    char input_string[100]; // Declare a string buffer with
a maximum length

    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input_string, sizeof(input_string), stdin); //
Read a line of input
    input_string[strcspn(input_string, "\n")] = 0; //
Remove trailing newline

    // Check if input is a valid string
    if (strlen(input_string) > 0) {
        // Call the function and display the result
        int number_of_vowels = count_vowels(input_string);
        printf("Number of vowels in '%s': %d\n",
input_string, number_of_vowels);
    } else {
        printf("Please enter a valid string.\n");
    }

    return 0;
}

/*
Example Output:
Enter a string: asd
Number of vowels in 'asd': 1
*/
```

40. Check Whether a String Starts and Ends with Certain Characters

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

// Function to prompt the user for input
void prompt_user(const char *message, char *input, size_t
size) {
    printf("%s", message);
    fgets(input, size, stdin);
    input[strcspn(input, "\n")] = 0; // Remove trailing
newline
}

int main() {
    char input_string[100]; // Buffer for the main string
    char start_char[10];    // Buffer for starting
characters
    char end_char[10];      // Buffer for ending characters

    // Prompt user for a string
    printf("Enter a string: ");
    fgets(input_string, sizeof(input_string), stdin);
    input_string[strcspn(input_string, "\n")] = 0; //
Remove trailing newline

    // Check if the input string is valid
    if (strlen(input_string) == 0) {
        printf("Please enter a valid string.\n");
        return 1;
    }

    // Prompt user for starting and ending characters
    prompt_user("Enter the starting characters: ",
start_char, sizeof(start_char));
    prompt_user("Enter the ending characters: ", end_char,
sizeof(end_char));
```

```c
    // Check if the string starts with the specified
characters
    int starts_with = strncmp(input_string, start_char,
strlen(start_char)) == 0;
    // Check if the string ends with the specified
characters
    int ends_with = strlen(end_char) == 0 ||
strcmp(input_string + strlen(input_string) -
strlen(end_char), end_char) == 0;

    // Display the result
    if (starts_with && ends_with) {
        printf("The string '%s' starts with '%s' and ends
with '%s'.\n", input_string, start_char, end_char);
    } else {
        printf("The string '%s' does not start with '%s' or
end with '%s'.\n", input_string, start_char, end_char);
    }

    return 0;
}

/*
Example Output:
Enter a string: asd
Enter the starting characters: a
Enter the ending characters: d
The string 'asd' starts with 'a' and ends with 'd'.
*/
```

41. Replace All Occurrences of a String

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

void replace_all_occurrences(const char *original, const
char *search, const char *replacement, char *result) {
    const char *pos = original;
    size_t search_len = strlen(search);
    size_t replacement_len = strlen(replacement);
    size_t result_index = 0;

    // Iterate through the original string
    while ((pos = strstr(pos, search)) != NULL) {
        // Copy the part of the original string before the
search string
        size_t length = pos - original;
        strncpy(result + result_index, original, length);
        result_index += length;

        // Copy the replacement string into the result
        strncpy(result + result_index, replacement,
replacement_len);
        result_index += replacement_len;

        // Move past the found occurrence
        pos += search_len;
        original = pos;
    }

    // Copy any remaining part of the original string
    strcpy(result + result_index, original);
}

int main() {
    // Example string
    const char *original_string = "Hello world, world!";
    // String to replace
    const char *search_string = "world";
    // Replacement string
    const char *replacement_string = "universe";

    // Create a buffer to hold the modified string
    char modified_string[256]; // Make sure this is large
enough to hold the result

    // Replace all occurrences of search_string with
replacement_string
    replace_all_occurrences(original_string, search_string,
replacement_string, modified_string);

    // Display the result
    printf("Original String: %s\n", original_string);
    printf("Modified String: %s\n", modified_string);

    return 0;
}

/*
Example Output:
Original String: Hello world, world!
Modified String: Hello universe, universe!
*/
```

42. Create Multiline Strings

```c
#include <stdio.h>

int main() {
    // Multiline string using escape sequences for new
lines
    const char *multiline_string =
        "This is a multiline string.\n"
        "It spans multiple lines.\n"
        "You can include line breaks and indentation
easily.\n";

    // Print the multiline string
    printf("%s", multiline_string);

    return 0;
}

/*
Example Output:
This is a multiline string.
It spans multiple lines.
You can include line breaks and indentation easily.
*/
```

43. Format Numbers as Currency Strings

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

// Function to format the number as a currency string
void format_currency(double amount, char *result) {
    // Split the number into integer and fractional parts
    long long integer_part = (long long)amount; // Get the
integer part
    long long fractional_part = (long long)((amount -
integer_part) * 100 + 0.5); // Get the fractional part,
rounded

    // Manually format the integer part with thousand
separators
    char formatted_integer[50]; // Buffer to hold the
formatted integer part
    int pos = 0;

    // Handle the sign for negative amounts
    if (integer_part < 0) {
        formatted_integer[pos++] = '-';
        integer_part = -integer_part; // Work with the
absolute value
    }

    // Add thousand separators
    char buffer[20];
    snprintf(buffer, sizeof(buffer), "%lld", integer_part);
// Convert integer to string

    int len = strlen(buffer);
    int count = 0;

    for (int i = len - 1; i >= 0; i--) {
        if (count > 0 && count % 3 == 0) {
            formatted_integer[pos++] = ','; // Add comma as
thousand separator
```

```c
        }
        formatted_integer[pos++] = buffer[i]; // Copy
character from the buffer
        count++;
    }

    // Reverse the formatted integer part
    for (int i = 0; i < pos / 2; i++) {
        char temp = formatted_integer[i];
        formatted_integer[i] = formatted_integer[pos - 1 -
i];
        formatted_integer[pos - 1 - i] = temp;
    }
    formatted_integer[pos] = '\0'; // Null-terminate the
string

    // Format the final currency string
    sprintf(result, "$%s.%02lld", formatted_integer,
fractional_part);
}

int main() {
    double amount = 1234567.89; // Example number
    char formatted_amount[100]; // Buffer to hold the
formatted currency string

    // Format the amount as a currency string
    format_currency(amount, formatted_amount);

    // Display the formatted currency string
    printf("Formatted Amount: %s\n", formatted_amount);

    return 0;
}

/*
Example Output:
Formatted Amount: $1,234,567.89
*/
```

44. Generate Random String

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

#define STRING_LENGTH 8

// Function to generate a random string of a given length
void generate_random_string(char *random_string, size_t
length) {
    const char characters[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345
6789";
    size_t num_characters = sizeof(characters) - 1; //
Exclude the null terminator

    // Seed the random number generator
    srand(time(NULL));

    for (size_t i = 0; i < length; i++) {
        int index = rand() % num_characters; // Get a
random index
        random_string[i] = characters[index]; // Assign a
random character
    }

    random_string[length] = '\0'; // Null-terminate the
string
}

int main() {
    char random_string[STRING_LENGTH + 1]; // +1 for null
terminator

    // Generate a random string of length 8
    generate_random_string(random_string, STRING_LENGTH);

    // Display the random string
    printf("Random String: %s\n", random_string);
    return 0;
}

/*
Example Output:
Random String: jQmNYWWC
*/
```

45. Check if a String Starts with Another String

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

int main() {
    // Example strings
    const char *main_string = "Hello, World!";
    const char *search_string = "Hello";

    // Check if main_string starts with search_string
    bool starts_with = strncmp(main_string, search_string,
strlen(search_string)) == 0;

    // Display the result
    printf("Does the string start with '%s'? %s\n",
search_string, starts_with ? "true" : "false");

    return 0;
}

/*
Example Output:
Does the string start with 'Hello'? true
*/
```

46. Trim a String

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

// Function to trim leading and trailing whitespace
void trim_whitespace(char *str) {
    // Trim leading whitespace
    while (isspace((unsigned char)*str)) str++;

    // Trim trailing whitespace
    char *end = str + strlen(str) - 1;
    while (end > str && isspace((unsigned char)*end)) end--
;

    // Null-terminate the trimmed string
    *(end + 1) = '\0';
}

int main() {
    // Example string with leading and trailing whitespaces
    char string_with_spaces[] = "   Hello, World!   ";

    // Trim the string
    trim_whitespace(string_with_spaces);

    // Display the result
    printf("Original String: '   Hello, World!   '\n");
    printf("Trimmed String: '%s'\n", string_with_spaces);

    return 0;
}

/*
Example Output:
Original String: '   Hello, World!   '
Trimmed String: 'Hello, World!'
*/
```