```
    unsigned long long lcm = (num1 * num2) / gcd(num1,
num2);
    cout << "The LCM of " << num1 << " and " << num2 << "
is: " << lcm << endl;

    return 0;
}

// Enter the first positive integer: 3
// Enter the second positive integer: 4
// The LCM of 3 and 4 is: 12
```

25. Find the Factors of a Number

This program finds and displays all factors of a positive integer entered by the user.

```
#include <iostream>
using namespace std;

// Function to read a positive number with input validation
unsigned long long read_positive_number(const string
&prompt) {
    unsigned long long num;
    cout << prompt;
    while (!(cin >> num) || num == 0) {
        cout << "Please enter a valid positive integer: ";
        cin.clear(); // clear error flag
        cin.ignore(numeric_limits<streamsize>::max(),
'\n'); // discard invalid input
    }
    return num;
}
int main() {
    unsigned long long number = read_positive_number("Enter
a positive integer: ");
    cout << "Factors of " << number << ":" << endl;

    // Find and display the factors
    for (unsigned long long i = 1; i <= number; i++) {
        if (number % i == 0) {
            cout << i << endl;
        }
    }

    return 0;
}
// Enter a positive integer: 3
// Factors of 3:
// 1
// 3
```

26. Find Sum of Natural Numbers Using Recursion

This program calculates the sum of the first `n` natural numbers using a recursive function.

```
#include <iostream>
using namespace std;

// Recursive function to calculate the sum of natural
numbers
unsigned long long sum_of_natural_numbers(unsigned long
long n) {
    if (n == 0) {
        return 0;
    }
    return n + sum_of_natural_numbers(n - 1);
}

// Function to read a positive number with input validation
unsigned long long read_positive_number(const string
&prompt) {
    unsigned long long num;
    cout << prompt;
    while (!(cin >> num) || num == 0) {
        cout << "Please enter a valid positive integer: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(),
'\n');
    }
    return num;
}

int main() {
    unsigned long long number = read_positive_number("Enter
a positive integer: ");
```

```
    cout << "The sum of natural numbers up to " << number
<< " is: "
         << sum_of_natural_numbers(number) << endl;

    return 0;
}

// Enter a positive integer: 5
// The sum of natural numbers up to 5 is: 15
```

27. Guess a Random Number

This program generates a random number between 1 and 100 and asks the user to guess it, giving hints whether the guess is too high or too low until the correct number is guessed.

```
#include <iostream>
#include <cstdlib>  // For rand() and srand()
#include <ctime>    // For time()
using namespace std;

// Function to read the user's guess with validation
unsigned int get_guess() {
    unsigned int guess;
    cout << "Guess the random number (1-100): ";
    if (!(cin >> guess)) {
        cin.clear(); // clear error flag
        cin.ignore(numeric_limits<streamsize>::max(),
'\n'); // discard invalid input
        return 0; // Return 0 for invalid input
    }
    return guess;
}

int main() {
    srand(time(NULL)); // Seed random number generator
    unsigned int random_number = rand() % 100 + 1; //
Random number between 1 and 100
    unsigned int attempts = 0;

    while (true) {
        unsigned int guess = get_guess();
        if (guess == 0) {
            cout << "Please enter a valid number." << endl;
            continue; // Prompt for valid guess
        }

        attempts++;
        if (guess < random_number) {
            cout << "Too low!" << endl;
```

```
        } else if (guess > random_number) {
            cout << "Too high!" << endl;
        } else {
            cout << "Congratulations! You guessed " <<
random_number
                 << " in " << attempts << " attempts." <<
endl;
            break; // Exit the loop
        }
    }

    return 0;
}

// Guess the random number (1-100): 50
// Too low!
// Guess the random number (1-100): 75
// Too high!
// Guess the random number (1-100): 63
// Congratulations! You guessed 63 in 3 attempts.
```

28. Shuffle Deck of Cards

This program creates a standard deck of 52 cards, prints it, shuffles the deck randomly, and prints the shuffled deck.

```
#include <iostream>
#include <string>
#include <cstdlib> // For rand() and srand()
#include <ctime>   // For time()
#include <algorithm> // For swap
using namespace std;

const int DECK_SIZE = 52;

// Structure for a card
struct Card {
    string rank; // Rank (2-10, Jack, Queen, King, Ace)
    string suit; // Suit (Hearts, Diamonds, Clubs, Spades)
};

// Function to create a deck of cards
void create_deck(Card deck[]) {
    string ranks[] = {"2", "3", "4", "5", "6", "7", "8",
"9", "10", "Jack", "Queen", "King", "Ace"};
    string suits[] = {"Hearts", "Diamonds", "Clubs",
"Spades"};

    int index = 0;
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 13; j++) {
            deck[index].rank = ranks[j];
            deck[index].suit = suits[i];
            index++;
        }
    }
}

// Function to shuffle the deck of cards
void shuffle_deck(Card deck[]) {
    for (int i = DECK_SIZE - 1; i > 0; i--) {
```

```
        int j = rand() % (i + 1); // Random index
        swap(deck[i], deck[j]);    // Swap cards
    }
}

// Function to print the deck of cards
void print_deck(const Card deck[]) {
    for (int i = 0; i < DECK_SIZE; i++) {
        cout << deck[i].rank << " of " << deck[i].suit <<
endl;
    }
}

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

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

    cout << "Initial Deck:" << endl;
    print_deck(deck);  // Print the initial deck

    shuffle_deck(deck); // Shuffle the deck

    cout << "\nShuffled Deck:" << endl;
    print_deck(deck);   // Print the shuffled deck

    return 0;
}

// Initial Deck:
// 2 of Hearts
// 3 of Hearts
// ...
// Shuffled Deck:
// Queen of Clubs
// 7 of Diamonds
// Ace of Spades
// ...
```

29. Display Fibonacci Sequence Using Recursion

This program prints the first `n` terms of the Fibonacci sequence using a recursive function.

```
#include <iostream>
using namespace std;

// 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;

    cout << "Enter the number of terms in the Fibonacci
sequence: ";
    if (!(cin >> num_terms)) {
        cout << "Please enter a valid non-negative
integer." << endl;
        return 1;
    }

    cout << "Fibonacci sequence of " << num_terms << "
terms:" << endl;
    for (unsigned int i = 0; i < num_terms; i++) {
        cout << fibonacci(i) << endl;
    }

    return 0;
}

// Enter the number of terms in the Fibonacci sequence: 6
// Fibonacci sequence of 6 terms:
// 0
```

```
// 1
// 1
// 2
// 3
// 5
```

30. Find Factorial of Number Using Recursion

This program calculates the factorial of a non-negative integer using a recursive function.

```
#include <iostream>
using namespace std;

// 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;

    cout << "Enter a non-negative integer: ";
    if (!(cin >> number) || number < 0) {
        cout << "Please enter a valid non-negative
integer." << endl;
        return 1;
    }

    cout << "The factorial of " << number << " is: " <<
factorial(number) << endl;

    return 0;
}

// Enter a non-negative integer: 5
// The factorial of 5 is: 120
```

31. Convert Decimal to Binary

This program converts a non-negative decimal number to its binary representation using an array to store digits.

```
#include <iostream>
using namespace std;

// Function to convert decimal to binary
void decimal_to_binary(unsigned int num) {
    if (num == 0) {
        cout << "The binary equivalent is: 0" << endl;
        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
    cout << "The binary equivalent is: ";
    for (int i = index - 1; i >= 0; i--) {
        cout << binary[i];
    }
    cout << endl;
}

int main() {
    unsigned int decimal_number;

    cout << "Enter a decimal number: ";
    if (!(cin >> decimal_number)) {
```

```
        cout << "Please enter a valid non-negative
integer." << endl;
        return 1;
    }

    decimal_to_binary(decimal_number);

    return 0;
}

// Enter a decimal number: 5
// The binary equivalent is: 101
```

32. Find ASCII Value of Character

This program reads a character from the user and displays its ASCII value.

```
#include <iostream>
using namespace std;

int main() {
    char character;

    cout << "Enter a character: ";
    cin >> character;  // Read a single character

    cout << "The ASCII value of '" << character << "' is: "
<< static_cast<int>(character) << endl;

    return 0;
}

/*
Example Runs:

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];
```

```c
    // 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

This program checks if a given string is a palindrome, ignoring case and non-alphanumeric characters.

```
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

// Function to clean the input string (remove non-
alphanumeric and convert to lowercase)
string clean_string(const string &input) {
    string cleaned;
    for (char ch : input) {
        if (isalnum(ch)) {
            cleaned += tolower(ch);
        }
    }
    return cleaned;
}

// Function to check if a string is a palindrome
bool is_palindrome(const string &str) {
    int left = 0, right = str.length() - 1;
    while (left < right) {
        if (str[left] != str[right]) return false;
        left++;
        right--;
    }
    return true;
}

int main() {
    string input;
    cout << "Enter a string: ";
    getline(cin, input);

    string cleaned = clean_string(input);
```

```
    if (is_palindrome(cleaned)) {
        cout << "\"" << input << "\" is a palindrome." <<
endl;
    } else {
        cout << "\"" << input << "\" is not a palindrome."
<< endl;
    }

    return 0;
}

/*
Example Runs:

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.
*/
```

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

This program reverses a string entered by the user.

```cpp
#include <iostream>
#include <string>
using namespace std;

// Function to reverse a string in place
void reverse_string(string &str) {
    int length = str.length();
    for (int i = 0; i < length / 2; i++) {
        swap(str[i], str[length - i - 1]);
    }
}

int main() {
    string input_string;

    // Prompt user for a string
    cout << "Enter a string: ";
    getline(cin, input_string);

    if (input_string.empty()) {
        cout << "Please enter a valid string." << endl;
        return 1;
    }

    // Reverse the string
    reverse_string(input_string);

    // Display the reversed string
    cout << "Reversed String: " << input_string << endl;

    return 0;
}

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

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

This program counts how many times a specific character appears in a user-provided string.

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string input_string;
    char target_char;
    int count = 0;

    // Prompt user for a string
    cout << "Enter a string: ";
    getline(cin, input_string);

    if (input_string.empty()) {
        cout << "Please enter a valid string." << endl;
        return 1;
    }

    // Prompt user for the character to count
    cout << "Enter the character to count: ";
    cin >> target_char;

    // Count occurrences of the target character
    for (char c : input_string) {
        if (c == target_char) {
            count++;
        }
    }

    // Display the result
    cout << "Number of occurrences of '" << target_char <<
"' in '" << input_string << "': " << count << endl;
```

```
    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

This program reads a string from the user and converts only the first character of the string to uppercase, leaving the rest of the string unchanged. It handles empty strings gracefully.

```
#include <iostream>
#include <string>
#include <cctype>  // For toupper
using namespace std;

// Function to capitalize the first letter
void capitalizeFirstLetter(string &str) {
    if (!str.empty()) {
        str[0] = toupper(str[0]);
    }
}

int main() {
    string input_string;

    // Prompt user for a string
    cout << "Enter a string: ";
    getline(cin, input_string);

    if (!input_string.empty()) {
        // Capitalize the first letter
        capitalizeFirstLetter(input_string);

        // Display the result
        cout << "String with First Letter Uppercase: " <<
input_string << endl;
    } else {
        cout << "Please enter a valid string." << endl;
    }

    return 0;
```

```
}

/*
Enter a string: hello
String with First Letter Uppercase: Hello

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

39. Count the Number of Vowels in a String

This program reads a string from the user and counts the total number of vowels (`a`, `e`, `i`, `o`, `u`) present in the string. It considers both uppercase and lowercase vowels and handles empty strings gracefully.

```
#include <iostream>
#include <string>
#include <cctype>

int count_vowels(const std::string &str) {
    int count = 0;
    for (char ch : str) {
        char lower_ch = std::tolower(ch); // Convert to
lowercase for uniformity
        if (lower_ch == 'a' || lower_ch == 'e' || lower_ch
== 'i' ||
            lower_ch == 'o' || lower_ch == 'u') {
            count++;
        }
    }
    return count;
}

int main() {
    std::string input_string;

    // Prompt user for a string
    std::cout << "Enter a string: ";
    std::getline(std::cin, input_string);

    // Check if input is valid
    if (!input_string.empty()) {
        int number_of_vowels = count_vowels(input_string);
        std::cout << "Number of vowels in '" <<
input_string << "': "
```

```
                << number_of_vowels << std::endl;
    } else {
        std::cout << "Please enter a valid string." <<
std::endl;
    }

    return 0;
}

/*
Enter a string: Hello World
Number of vowels in 'Hello World': 3
*/
```

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

This program reads a string from the user and checks whether it starts with a specified set of characters and ends with another specified set. The user provides both the starting and ending characters, and the program validates the string accordingly.

```
#include <iostream>
#include <string>

// Function to check if a string starts with a given prefix
bool starts_with(const std::string &str, const std::string
&prefix) {
    if (prefix.size() > str.size()) return false;
    return str.substr(0, prefix.size()) == prefix;
}

// Function to check if a string ends with a given suffix
bool ends_with(const std::string &str, const std::string
&suffix) {
    if (suffix.size() > str.size()) return false;
    return str.substr(str.size() - suffix.size()) ==
suffix;
}

int main() {
    std::string input_string, start_chars, end_chars;

    // Prompt user for main string
    std::cout << "Enter a string: ";
    std::getline(std::cin, input_string);

    if (input_string.empty()) {
        std::cout << "Please enter a valid string." <<
std::endl;
        return 1;
    }
```

```
    // Prompt user for starting and ending characters
    std::cout << "Enter the starting characters: ";
    std::getline(std::cin, start_chars);

    std::cout << "Enter the ending characters: ";
    std::getline(std::cin, end_chars);

    // Check start and end
    if (starts_with(input_string, start_chars) &&
ends_with(input_string, end_chars)) {
        std::cout << "The string '" << input_string
                  << "' starts with '" << start_chars
                  << "' and ends with '" << end_chars <<
"'." << std::endl;
    } else {
        std::cout << "The string '" << input_string
                  << "' does not start with '" <<
start_chars
                  << "' or end with '" << end_chars << "'."
<< std::endl;
    }

    return 0;
}

/*
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

This program searches for all occurrences of a specific substring within a string and replaces them with another substring provided by the user. It handles multiple occurrences and constructs a new modified string.

```
#include <iostream>
#include <string>

// Function to replace all occurrences of a substring in a
string
std::string replace_all_occurrences(const std::string
&original, const std::string &search, const std::string
&replacement) {
    std::string result = original;
    size_t pos = 0;

    // Loop until no more occurrences are found
    while ((pos = result.find(search, pos)) !=
std::string::npos) {
        result.replace(pos, search.length(), replacement);
        pos += replacement.length(); // Move past the
replacement
    }

    return result;
}

int main() {
    std::string original_string = "Hello world, world!";
    std::string search_string = "world";
    std::string replacement_string = "universe";

    // Replace all occurrences
    std::string modified_string =
replace_all_occurrences(original_string, search_string,
replacement_string);
```

```
    // Display results
    std::cout << "Original String: " << original_string <<
std::endl;
    std::cout << "Modified String: " << modified_string <<
std::endl;

    return 0;
}

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