**EXTRA LAB EXERCISES FOR IMPROVING PROGRAMMING LOGIC**

**1. Operators**

**1.Write a C program that acts as a simple calculator. The program should take two numbers and an operator as input from the user and perform the respective operation (addition, subtraction, multiplication, division, or modulus) using operators.**

**Challenge: Extend the program to handle invalid operator inputs.**

#include <stdio.h>

int main() {

double num1, num2, result;

char operator;

// Prompt the user for input

printf("Enter first number: ");

scanf("%lf", &num1);

printf("Enter operator (+, -, \*, /, %): ");

scanf(" %c", &operator); // Space before %c to handle any leftover newline characters

printf("Enter second number: ");

scanf("%lf", &num2);

// Perform the operation based on the operator input

switch(operator) {

case '+':

result = num1 + num2;

printf("%.2lf %c %.2lf = %.2lf\n", num1, operator, num2, result);

break;

case '-':

result = num1 - num2;

printf("%.2lf %c %.2lf = %.2lf\n", num1, operator, num2, result);

break;

case '\*':

result = num1 \* num2;

printf("%.2lf %c %.2lf = %.2lf\n", num1, operator, num2, result);

break;

case '/':

if (num2 != 0) {

result = num1 / num2;

printf("%.2lf %c %.2lf = %.2lf\n", num1, operator, num2, result);

} else {

printf("Error: Division by zero is not allowed.\n");

}

break;

case '%':

if ((int)num2 != 0) {

result = (int)num1 % (int)num2;

printf("%d %c %d = %d\n", (int)num1, operator, (int)num2, (int)result);

} else {

printf("Error: Modulus by zero is not allowed.\n");

}

break;

default:

printf("Error: Invalid operator.\n");

break;

}

return 0;

}

**2. Write a C program that takes an integer from the user and checks the following using different operators:**

**o Whether the number is even or odd.**

**o Whether the number is positive, negative, or zero.**

**o Whether the number is a multiple of both 3 and 5.**

#include <stdio.h>

int main() {

int num;

// Prompt the user to enter an integer

printf("Enter an integer: ");

scanf("%d", &num);

// Check if the number is even or odd

if (num % 2 == 0) {

printf("%d is an even number.\n", num);

} else {

printf("%d is an odd number.\n", num);

}

// Check if the number is positive, negative, or zero

if (num > 0) {

printf("%d is a positive number.\n", num);

} else if (num < 0) {

printf("%d is a negative number.\n", num);

} else {

printf("%d is zero.\n", num);

}

// Check if the number is a multiple of both 3 and 5

if (num % 3 == 0 && num % 5 == 0) {

printf("%d is a multiple of both 3 and 5.\n", num);

} else {

printf("%d is not a multiple of both 3 and 5.\n", num);

}

return 0;

}

**2. Control Statements**

**3. Write a C program that takes the marks of a student as input and displays the corresponding grade based on the following conditions:**

**o Marks > 90: Grade A**

**o Marks > 75 and <= 90: Grade B**

**o Marks > 50 and <= 75: Grade**

**C o Marks <= 50: Grade D**

**Use if-else or switch statements for the decision-making process.**

#include <stdio.h>

int main() {

int marks;

// Taking input for marks

printf("Enter the marks of the student: ");

scanf("%d", &marks);

// Decision-making based on the marks

if (marks > 90) {

printf("Grade A\n");

} else if (marks > 75 && marks <= 90) {

printf("Grade B\n");

} else if (marks > 50 && marks <= 75) {

printf("Grade C\n");

} else if (marks <= 50) {

printf("Grade D\n");

} else {

printf("Invalid input.\n");

}

return 0;

}

**4. Write a C program that takes three numbers from the user and determines:**

**o The largest number.**

**o The smallest number.**

**Challenge: Solve the problem using both if-else and switch-case statements.**

Solution using if-else:

#include <stdio.h>

int main() {

int num1, num2, num3;

// Taking input of three numbers

printf("Enter three numbers: ");

scanf("%d %d %d", &num1, &num2, &num3);

// Finding the largest number using if-else

if (num1 >= num2 && num1 >= num3) {

printf("Largest number is: %d\n", num1);

} else if (num2 >= num1 && num2 >= num3) {

printf("Largest number is: %d\n", num2);

} else {

printf("Largest number is: %d\n", num3);

}

// Finding the smallest number using if-else

if (num1 <= num2 && num1 <= num3) {

printf("Smallest number is: %d\n", num1);

} else if (num2 <= num1 && num2 <= num3) {

printf("Smallest number is: %d\n", num2);

} else {

printf("Smallest number is: %d\n", num3);

}

return 0;

}

Solution using switch-case:

#include <stdio.h>

int main() {

int num1, num2, num3;

int largest, smallest;

// Taking input of three numbers

printf("Enter three numbers: ");

scanf("%d %d %d", &num1, &num2, &num3);

// Finding the largest number using switch-case

switch (1) {

case 1:

if (num1 >= num2 && num1 >= num3) {

largest = num1;

} else if (num2 >= num1 && num2 >= num3) {

largest = num2;

} else {

largest = num3;

}

break;

}

// Finding the smallest number using switch-case

switch (1) {

case 1:

if (num1 <= num2 && num1 <= num3) {

smallest = num1;

} else if (num2 <= num1 && num2 <= num3) {

smallest = num2;

} else {

smallest = num3;

}

break;

}

// Output the results

printf("Largest number is: %d\n", largest);

printf("Smallest number is: %d\n", smallest);

return 0;

}

**3. Loops**

**5. Write a C program that checks whether a given number is a prime number or not using a for loop.**

**Challenge: Modify the program to print all prime numbers between 1 and a given number.**

Check if a given number is a prime number:

#include <stdio.h>

int main() {

int num, i, isPrime = 1;

// Taking input from the user

printf("Enter a number to check if it is prime: ");

scanf("%d", &num);

// Check if the number is less than 2 (not prime)

if (num <= 1) {

printf("%d is not a prime number.\n", num);

return 0;

}

// Check for factors of the number

for (i = 2; i <= num / 2; i++) {

if (num % i == 0) {

isPrime = 0; // num is divisible by i, so it is not prime

break;

}

}

// Output whether the number is prime or not

if (isPrime) {

printf("%d is a prime number.\n", num);

} else {

printf("%d is not a prime number.\n", num);

}

return 0;

}

Print all prime numbers between 1 and a given number:

#include <stdio.h>

int main() {

int limit, num, i, j, isPrime;

// Taking input for the limit

printf("Enter a number to print all prime numbers between 1 and that number: ");

scanf("%d", &limit);

printf("Prime numbers between 1 and %d are:\n", limit);

// Loop through all numbers from 2 to the limit

for (num = 2; num <= limit; num++) {

isPrime = 1; // Assume the number is prime

// Check if the number is divisible by any number from 2 to num/2

for (i = 2; i <= num / 2; i++) {

if (num % i == 0) {

isPrime = 0; // num is not prime

break;

}

}

// If the number is prime, print it

if (isPrime) {

printf("%d ", num);

}

}

printf("\n");

return 0;

}

**6. Write a C program that takes an integer input from the user and prints its multiplication table using a for loop.**

**Challenge: Allow the user to input the range of the multiplication table (e.g., from 1 to N).**

#include <stdio.h>

int main() {

int num, range, i;

// Taking input for the number and range of the multiplication table

printf("Enter the number for which you want the multiplication table: ");

scanf("%d", &num);

printf("Enter the range for the multiplication table (e.g., up to N): ");

scanf("%d", &range);

// Printing the multiplication table from 1 to N

printf("Multiplication table of %d (from 1 to %d):\n", num, range);

for (i = 1; i <= range; i++) {

printf("%d x %d = %d\n", num, i, num \* i);

}

return 0;

}

**7. Write a C program that takes an integer from the user and calculates the sum of its digits using a while loop.**

**Challenge: Extend the program to reverse the digits of the number.**

#include <stdio.h>

int main() {

int num, sum = 0, reverse = 0, digit;

// Taking input from the user

printf("Enter an integer: ");

scanf("%d", &num);

// Store the original number to calculate the sum and reverse it

int originalNum = num;

// While loop to calculate the sum of digits and reverse the number

while (num != 0) {

// Get the last digit

digit = num % 10;

// Add the digit to the sum

sum += digit;

// Reverse the digits

reverse = reverse \* 10 + digit;

// Remove the last digit from the number

num /= 10;

}

// Output the sum of digits and the reversed number

printf("Sum of the digits of %d: %d\n", originalNum, sum);

printf("Reversed number: %d\n", reverse);

return 0;

}

**4. Arrays**

**8. Write a C program that accepts 10 integers from the user and stores them in an array. The program should then find and print the maximum and minimum values in the array.**

**Challenge: Extend the program to sort the array in ascending order.**

#include <stdio.h>

int main() {

int arr[10], i, j, temp;

int max, min;

// Accept 10 integers from the user

printf("Enter 10 integers:\n");

for (i = 0; i < 10; i++) {

scanf("%d", &arr[i]);

}

// Find the maximum and minimum values

max = arr[0];

min = arr[0];

for (i = 1; i < 10; i++) {

if (arr[i] > max) {

max = arr[i];

}

if (arr[i] < min) {

min = arr[i];

}

}

// Print the maximum and minimum values

printf("Maximum value: %d\n", max);

printf("Minimum value: %d\n", min);

// Sort the array in ascending order using Bubble Sort

for (i = 0; i < 10 - 1; i++) {

for (j = 0; j < 10 - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

// Swap the elements

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

// Print the sorted array

printf("Sorted array in ascending order: ");

for (i = 0; i < 10; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**9. Write a C program that accepts two 2x2 matrices from the user and adds them. Display the resultant matrix.**

**Challenge: Extend the program to work with 3x3 matrices and matrix multiplication.**

Adding Two 2x2 Matrices

#include <stdio.h>

int main() {

int A[2][2], B[2][2], sum[2][2];

int i, j;

// Input for matrix A

printf("Enter elements for matrix A (2x2):\n");

for (i = 0; i < 2; i++) {

for (j = 0; j < 2; j++) {

printf("A[%d][%d]: ", i + 1, j + 1);

scanf("%d", &A[i][j]);

}

}

// Input for matrix B

printf("Enter elements for matrix B (2x2):\n");

for (i = 0; i < 2; i++) {

for (j = 0; j < 2; j++) {

printf("B[%d][%d]: ", i + 1, j + 1);

scanf("%d", &B[i][j]);

}

}

// Matrix addition

for (i = 0; i < 2; i++) {

for (j = 0; j < 2; j++) {

sum[i][j] = A[i][j] + B[i][j];

}

}

// Display the resultant matrix

printf("\nSum of the matrices A and B is:\n");

for (i = 0; i < 2; i++) {

for (j = 0; j < 2; j++) {

printf("%d ", sum[i][j]);

}

printf("\n");

}

return 0;

}

Extending to 3x3 Matrices and Matrix Multiplication

#include <stdio.h>

int main() {

int A[3][3], B[3][3], sum[3][3], mul[3][3];

int i, j, k;

// Input for matrix A (3x3)

printf("Enter elements for matrix A (3x3):\n");

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

printf("A[%d][%d]: ", i + 1, j + 1);

scanf("%d", &A[i][j]);

}

}

// Input for matrix B (3x3)

printf("Enter elements for matrix B (3x3):\n");

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

printf("B[%d][%d]: ", i + 1, j + 1);

scanf("%d", &B[i][j]);

}

}

// Matrix addition (3x3)

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

sum[i][j] = A[i][j] + B[i][j];

}

}

// Matrix multiplication (3x3)

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

mul[i][j] = 0;

for (k = 0; k < 3; k++) {

mul[i][j] += A[i][k] \* B[k][j];

}

}

}

// Display the resultant sum matrix

printf("\nSum of matrices A and B is:\n");

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

printf("%d ", sum[i][j]);

}

printf("\n");

}

// Display the resultant multiplication matrix

printf("\nMultiplication of matrices A and B is:\n");

for (i = 0; i < 3; i++) {

for (j = 0; j < 3; j++) {

printf("%d ", mul[i][j]);

}

printf("\n");

}

return 0;

}

**10. Write a C program that takes N numbers from the user and stores them in an array. The program should then calculate and display the sum of all array elements.**

**Challenge: Modify the program to also find the average of the numbers.**

#include <stdio.h>

int main() {

int N, i, sum = 0;

float average;

// Taking input for the size of the array (N numbers)

printf("Enter the number of elements (N): ");

scanf("%d", &N);

int arr[N];

// Taking N numbers as input from the user

printf("Enter %d numbers:\n", N);

for (i = 0; i < N; i++) {

scanf("%d", &arr[i]);

sum += arr[i]; // Calculating sum while inputting the numbers

}

// Calculating the average

average = (float)sum / N;

// Displaying the sum and average

printf("Sum of all array elements: %d\n", sum);

printf("Average of the numbers: %.2f\n", average);

return 0;

}

**5. Functions**

**11. Write a C program that generates the Fibonacci sequence up to N terms using a recursive function.**

**Challenge: Modify the program to calculate the Nth Fibonacci number using both iterative and recursive methods. Compare their efficiency.**

#include <stdio.h>

#include <time.h>

// Recursive function to calculate Fibonacci number

long long int fibonacci\_recursive(int n) {

if (n <= 1) {

return n;

}

return fibonacci\_recursive(n - 1) + fibonacci\_recursive(n - 2);

}

// Iterative function to calculate Fibonacci number

long long int fibonacci\_iterative(int n) {

long long int a = 0, b = 1, next;

if (n == 0) return a;

if (n == 1) return b;

for (int i = 2; i <= n; i++) {

next = a + b;

a = b;

b = next;

}

return b;

}

int main() {

int N;

// Taking input for the number of terms in the Fibonacci sequence

printf("Enter the value of N: ");

scanf("%d", &N);

// Printing the Fibonacci sequence up to N terms using recursion

printf("\nFibonacci sequence up to %d terms (using recursion):\n", N);

for (int i = 0; i < N; i++) {

printf("%lld ", fibonacci\_recursive(i));

}

printf("\n");

// Measuring the time taken for the recursive method to calculate the Nth Fibonacci number

clock\_t start\_recursive = clock();

long long int fib\_recursive = fibonacci\_recursive(N - 1);

clock\_t end\_recursive = clock();

double time\_taken\_recursive = (double)(end\_recursive - start\_recursive) / CLOCKS\_PER\_SEC;

// Printing the Nth Fibonacci number using recursion

printf("\nNth Fibonacci number (using recursion): %lld\n", fib\_recursive);

printf("Time taken for recursive method: %f seconds\n", time\_taken\_recursive);

// Measuring the time taken for the iterative method to calculate the Nth Fibonacci number

clock\_t start\_iterative = clock();

long long int fib\_iterative = fibonacci\_iterative(N - 1);

clock\_t end\_iterative = clock();

double time\_taken\_iterative = (double)(end\_iterative - start\_iterative) / CLOCKS\_PER\_SEC;

// Printing the Nth Fibonacci number using iteration

printf("\nNth Fibonacci number (using iteration): %lld\n", fib\_iterative);

printf("Time taken for iterative method: %f seconds\n", time\_taken\_iterative);

// Comparing the efficiency

if (time\_taken\_recursive > time\_taken\_iterative) {

printf("\nThe iterative method is more efficient.\n");

} else if (time\_taken\_recursive < time\_taken\_iterative) {

printf("\nThe recursive method is more efficient.\n");

} else {

printf("\nBoth methods took the same time.\n");

}

return 0;

}

**12. Write a C program that calculates the factorial of a given number using a function.**

**Challenge: Implement both an iterative and a recursive version of the factorial function and compare their performance for large numbers.**

#include <stdio.h>

#include <time.h>

// Iterative function to calculate factorial

long long int factorial\_iterative(int n) {

long long int result = 1;

for (int i = 1; i <= n; i++) {

result \*= i;

}

return result;

}

// Recursive function to calculate factorial

long long int factorial\_recursive(int n) {

if (n == 0 || n == 1) {

return 1;

}

return n \* factorial\_recursive(n - 1);

}

int main() {

int num;

// Taking input for the number to calculate factorial

printf("Enter a number to calculate its factorial: ");

scanf("%d", &num);

// Measuring the time taken for the iterative method

clock\_t start\_iterative = clock();

long long int fact\_iterative = factorial\_iterative(num);

clock\_t end\_iterative = clock();

double time\_taken\_iterative = (double)(end\_iterative - start\_iterative) / CLOCKS\_PER\_SEC;

// Printing the result of iterative factorial calculation

printf("\nFactorial of %d (using iterative method): %lld\n", num, fact\_iterative);

printf("Time taken for iterative method: %f seconds\n", time\_taken\_iterative);

// Measuring the time taken for the recursive method

clock\_t start\_recursive = clock();

long long int fact\_recursive = factorial\_recursive(num);

clock\_t end\_recursive = clock();

double time\_taken\_recursive = (double)(end\_recursive - start\_recursive) / CLOCKS\_PER\_SEC;

// Printing the result of recursive factorial calculation

printf("\nFactorial of %d (using recursive method): %lld\n", num, fact\_recursive);

printf("Time taken for recursive method: %f seconds\n", time\_taken\_recursive);

// Comparing the performance of both methods

if (time\_taken\_recursive > time\_taken\_iterative) {

printf("\nThe iterative method is more efficient for this calculation.\n");

} else if (time\_taken\_recursive < time\_taken\_iterative) {

printf("\nThe recursive method is more efficient for this calculation.\n");

} else {

printf("\nBoth methods took the same amount of time.\n");

}

return 0;

}

**13. Write a C program that takes a number as input and checks whether it is a palindrome using a function.**

**Challenge: Modify the program to check if a given string is a palindrome.**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

// Function to check if a number is a palindrome

int is\_palindrome\_number(int n) {

int original = n, reversed = 0, remainder;

// Reverse the number

while (n != 0) {

remainder = n % 10;

reversed = reversed \* 10 + remainder;

n /= 10;

}

// Check if the original number and the reversed number are the same

return original == reversed;

}

// Function to check if a string is a palindrome

int is\_palindrome\_string(char str[]) {

int start = 0, end = strlen(str) - 1;

// Check each character from start and end

while (start < end) {

// Skip non-alphanumeric characters and make the comparison case-insensitive

if (!isalnum(str[start])) {

start++;

} else if (!isalnum(str[end])) {

end--;

} else if (tolower(str[start]) != tolower(str[end])) {

return 0; // Not a palindrome

} else {

start++;

end--;

}

}

return 1; // Is a palindrome

}

int main() {

int choice, num;

char str[100];

// Menu for the user to choose between checking a number or string palindrome

printf("Enter 1 to check if a number is a palindrome\n");

printf("Enter 2 to check if a string is a palindrome\n");

printf("Choice: ");

scanf("%d", &choice);

if (choice == 1) {

// Check if the number is a palindrome

printf("Enter a number: ");

scanf("%d", &num);

if (is\_palindrome\_number(num)) {

printf("%d is a palindrome number.\n", num);

} else {

printf("%d is not a palindrome number.\n", num);

}

} else if (choice == 2) {

// Check if the string is a palindrome

printf("Enter a string: ");

getchar(); // Consume the newline character left by the previous scanf

fgets(str, sizeof(str), stdin);

// Remove the newline character from the string

str[strcspn(str, "\n")] = '\0';

if (is\_palindrome\_string(str)) {

printf("\"%s\" is a palindrome string.\n", str);

} else {

printf("\"%s\" is not a palindrome string.\n", str);

}

} else {

printf("Invalid choice!\n");

}

return 0;

}

**6. Strings**

**14. Write a C program that takes a string as input and reverses it using a function.**

**Challenge: Write the program without using built-in string handling functions.**

#include <stdio.h>

// Function to reverse a string without using built-in string functions

void reverse\_string(char str[]) {

int start = 0, end = 0;

// Finding the length of the string manually

while (str[end] != '\0') {

end++;

}

end--; // To point to the last valid character

// Swap characters from start to end until the middle of the string is reached

while (start < end) {

// Swap characters

char temp = str[start];

str[start] = str[end];

str[end] = temp;

// Move pointers towards the center

start++;

end--;

}

}

int main() {

char str[100];

// Taking input string from the user

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

// Remove the trailing newline character if present

int i = 0;

while (str[i] != '\0') {

if (str[i] == '\n') {

str[i] = '\0';

break;

}

i++;

}

// Calling the function to reverse the string

reverse\_string(str);

// Printing the reversed string

printf("Reversed string: %s\n", str);

return 0;

}

**15. Write a C program that takes a string from the user and counts the number of vowels and consonants in the string.**

**Challenge: Extend the program to also count digits and special characters.**

#include <stdio.h>

#include <ctype.h> // For the tolower() function

// Function to count vowels, consonants, digits, and special characters

void count\_characters(char str[], int \*vowels, int \*consonants, int \*digits, int \*special\_characters) {

\*vowels = 0;

\*consonants = 0;

\*digits = 0;

\*special\_characters = 0;

// Loop through each character of the string

for (int i = 0; str[i] != '\0'; i++) {

// Convert to lowercase to simplify the check

char ch = tolower(str[i]);

// Check if the character is a vowel

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

(\*vowels)++;

}

// Check if the character is a consonant (alphabetic character)

else if ((ch >= 'a' && ch <= 'z')) {

(\*consonants)++;

}

// Check if the character is a digit

else if (ch >= '0' && ch <= '9') {

(\*digits)++;

}

// If it is neither a vowel, consonant, nor a digit, it's a special character

else {

(\*special\_characters)++;

}

}

}

int main() {

char str[100];

int vowels, consonants, digits, special\_characters;

// Taking input string from the user

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

// Remove the trailing newline character if present

int i = 0;

while (str[i] != '\0') {

if (str[i] == '\n') {

str[i] = '\0';

break;

}

i++;

}

// Call the function to count vowels, consonants, digits, and special characters

count\_characters(str, &vowels, &consonants, &digits, &special\_characters);

// Output the counts

printf("\nNumber of vowels: %d\n", vowels);

printf("Number of consonants: %d\n", consonants);

printf("Number of digits: %d\n", digits);

printf("Number of special characters: %d\n", special\_characters);

return 0;

}

**16. Write a C program that counts the number of words in a sentence entered by the user.**

**Challenge: Modify the program to find the longest word in the sentence.**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

// Function to count words in a sentence

int count\_words(char str[]) {

int count = 0, i = 0;

int in\_word = 0;

// Loop through each character of the string

while (str[i] != '\0') {

if (isalnum(str[i])) {

// We are inside a word

if (!in\_word) {

count++;

in\_word = 1; // Start a new word

}

} else {

in\_word = 0; // End of word

}

i++;

}

return count;

}

// Function to find the longest word in a sentence

void find\_longest\_word(char str[], char longest\_word[]) {

int max\_length = 0, current\_length = 0, i = 0, j = 0;

int start = -1;

// Loop through the string to find the longest word

while (str[i] != '\0') {

if (isalnum(str[i])) {

// If we're inside a word, count its length

if (current\_length == 0) {

start = i; // Mark the start of the word

}

current\_length++;

} else {

// End of a word, check if it's the longest

if (current\_length > max\_length) {

max\_length = current\_length;

// Copy the longest word found so far

for (int k = 0; k < max\_length; k++) {

longest\_word[k] = str[start + k];

}

longest\_word[max\_length] = '\0'; // Null-terminate the string

}

current\_length = 0; // Reset the current word length

}

i++;

}

// Check the last word after the loop

if (current\_length > max\_length) {

max\_length = current\_length;

for (int k = 0; k < max\_length; k++) {

longest\_word[k] = str[start + k];

}

longest\_word[max\_length] = '\0';

}

}

int main() {

char str[200], longest\_word[100];

// Taking input from the user

printf("Enter a sentence: ");

fgets(str, sizeof(str), stdin);

// Remove the newline character if present

int i = 0;

while (str[i] != '\0') {

if (str[i] == '\n') {

str[i] = '\0';

break;

}

i++;

}

// Counting the number of words in the sentence

int word\_count = count\_words(str);

printf("Number of words: %d\n", word\_count);

// Finding the longest word

find\_longest\_word(str, longest\_word);

printf("Longest word: %s\n", longest\_word);

return 0;

}

**Extra Logic Building Challenges**

**17. Write a C program that checks whether a given number is an Armstrong number or not (e.g., 153 = 1^3 + 5^3 + 3^3).**

**Challenge: Write a program to find all Armstrong numbers between 1 and 1000.**

#include <stdio.h>

#include <math.h>

// Function to check if a number is an Armstrong number

int is\_armstrong(int num) {

int original\_num, remainder, result = 0, n = 0;

original\_num = num;

// Finding the number of digits

while (original\_num != 0) {

original\_num /= 10;

++n;

}

original\_num = num;

// Checking if sum of digits raised to the power of n is equal to the original number

while (original\_num != 0) {

remainder = original\_num % 10;

result += pow(remainder, n);

original\_num /= 10;

}

// If the sum of powers equals the original number, it is an Armstrong number

return result == num;

}

int main() {

int num;

// Input a number to check if it's an Armstrong number

printf("Enter a number to check if it is an Armstrong number: ");

scanf("%d", &num);

if (is\_armstrong(num)) {

printf("%d is an Armstrong number.\n", num);

} else {

printf("%d is not an Armstrong number.\n", num);

}

// Challenge: Find all Armstrong numbers between 1 and 1000

printf("Armstrong numbers between 1 and 1000 are:\n");

for (int i = 1; i <= 1000; i++) {

if (is\_armstrong(i)) {

printf("%d ", i);

}

}

printf("\n");

return 0;

}

**18. Write a C program that generates Pascal’s Triangle up to N rows using loops.**

**Challenge: Implement the same program using a recursive function.**

Using Loops:

#include <stdio.h>

// Function to generate Pascal's Triangle using loops

void generate\_pascals\_triangle(int n) {

int i, j, number;

// Loop for each row

for (i = 0; i < n; i++) {

// Print leading spaces for formatting

for (j = 0; j < n - i - 1; j++) {

printf(" ");

}

// Print the elements of the row

number = 1; // The first element of each row is always 1

for (j = 0; j <= i; j++) {

printf("%d ", number);

number = number \* (i - j) / (j + 1); // Using the property of binomial coefficients

}

printf("\n");

}

}

int main() {

int n;

// Take input for number of rows

printf("Enter the number of rows for Pascal's Triangle: ");

scanf("%d", &n);

// Generate Pascal's Triangle

generate\_pascals\_triangle(n);

return 0;

}

Using Recursion:

#include <stdio.h>

// Function to calculate binomial coefficient C(n, k)

int binomial\_coefficient(int n, int k) {

if (k == 0 || k == n) {

return 1;

}

return binomial\_coefficient(n - 1, k - 1) + binomial\_coefficient(n - 1, k);

}

// Function to print a row of Pascal's Triangle

void print\_pascals\_row(int n, int row) {

if (row > n) {

return;

}

// Print the number at position (n, row)

printf("%d ", binomial\_coefficient(n, row));

print\_pascals\_row(n, row + 1); // Recursive call for the next element in the row

}

// Function to print Pascal's Triangle up to N rows

void generate\_pascals\_triangle(int n, int current\_row) {

if (current\_row == n) {

return;

}

// Print leading spaces to center-align the triangle

for (int i = 0; i < n - current\_row - 1; i++) {

printf(" ");

}

// Print the current row of Pascal's Triangle

print\_pascals\_row(current\_row, 0);

printf("\n");

// Recursive call for the next row

generate\_pascals\_triangle(n, current\_row + 1);

}

int main() {

int n;

// Take input for the number of rows

printf("Enter the number of rows for Pascal's Triangle: ");

scanf("%d", &n);

// Generate Pascal's Triangle using recursion

generate\_pascals\_triangle(n, 0);

return 0;

}

**19. Write a C program that implements a simple number guessing game. The program should generate a random number between 1 and 100, and the user should guess the number within a limited number of attempts.**

**Challenge: Provide hints to the user if the guessed number is too high or too low.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define MAX\_ATTEMPTS 10 // Maximum number of attempts

// Function to start the number guessing game

void number\_guessing\_game() {

int target\_number, user\_guess;

int attempts = 0;

// Seed the random number generator

srand(time(0));

// Generate a random number between 1 and 100

target\_number = rand() % 100 + 1;

printf("Welcome to the Number Guessing Game!\n");

printf("I have selected a number between 1 and 100.\n");

printf("You have %d attempts to guess the number.\n", MAX\_ATTEMPTS);

// Loop for the user to make guesses

while (attempts < MAX\_ATTEMPTS) {

printf("Attempt %d: Enter your guess: ", attempts + 1);

scanf("%d", &user\_guess);

attempts++;

// Check if the guess is correct

if (user\_guess == target\_number) {

printf("Congratulations! You've guessed the correct number: %d\n", target\_number);

break;

}

// Provide hints if the guess is too high or too low

else if (user\_guess > target\_number) {

printf("Too high! Try again.\n");

} else {

printf("Too low! Try again.\n");

}

// Check if the user has exhausted all attempts

if (attempts == MAX\_ATTEMPTS) {

printf("Sorry, you've used all your attempts. The correct number was: %d\n", target\_number);

}

}

}

int main() {

number\_guessing\_game(); // Start the guessing game

return 0;

}