1. Operators:

LAB EXERCISE 1: Simple Calculator

1.1 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.

#include <stdio.h>

int main() {

double num1, num2, result;

char operator;

printf("Enter first number: ");

scanf("%lf", &num1);

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

scanf(" %c", &operator);

printf("Enter second number: ");

scanf("%lf", &num2);

switch (operator) {

case '+':

result = num1 + num2;

printf("Result: %.2lf\n", result);

break;

case '-':

result = num1 - num2;

printf("Result: %.2lf\n", result);

break;

case '\*':

result = num1 \* num2;

printf("Result: %.2lf\n", result);

break;

case '/':

if (num2 != 0) {

result = num1 / num2;

printf("Result: %.2lf\n", result);

} else {

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

}

break;

case '%':

if ((int)num2 != 0) {

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

printf("Result: %.0lf\n", result);

} else {

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

}

break;

default:

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

}

return 0;

}

Challenge: Extend the program to handle invalid operator inputs.

Ans.

#include <stdio.h>

int main() {

double num1, num2, result;

char operator;

printf("Enter first number: ");

scanf("%lf", &num1);

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

scanf(" %c", &operator);

printf("Enter second number: ");

scanf("%lf", &num2);

if (operator != '+' && operator != '-' && operator != '\*' && operator != '/' && operator != '%') {

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

} else {

switch (operator) {

case '+':

result = num1 + num2;

printf("Result: %.2lf\n", result);

break;

case '-':

result = num1 - num2;

printf("Result: %.2lf\n", result);

break;

case '\*':

result = num1 \* num2;

printf("Result: %.2lf\n", result);

break;

case '/':

if (num2 != 0) {

result = num1 / num2;

printf("Result: %.2lf\n", result);

} else {

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

}

break;

case '%':

if ((int)num2 != 0) {

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

printf("Result: %.0lf\n", result);

} else {

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

}

break;

}

}

return 0;

}

LAB EXERCISE 2: Check Number Properties

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

1. Whether the number is even or odd.

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

if (num % 2 == 0) {

printf("The number is even.\n");

} else {

printf("The number is odd.\n");

}

return 0;

}

2. Whether the number is positive, negative, or zero.

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

if (num > 0) {

printf("The number is positive.\n");

} else if (num < 0) {

printf("The number is negative.\n");

} else {

printf("The number is zero.\n");

}

return 0;

}

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

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

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

printf("The number is a multiple of both 3 and 5.\n");

} else {

printf("The number is not a multiple of both 3 and 5.\n");

}

return 0;

}

2. Control Statements

LAB EXERCISE 1: Grade Calculator

• 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;

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

scanf("%d", &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 {

printf("Grade D\n");

}

return 0;

}

LAB EXERCISE 2: Number Comparison

2.1 Write a C program that takes three numbers from the user and determines:

The largest number.

The smallest number

#include <stdio.h>

int main() {

int num1, num2, num3;

printf("Enter three numbers: ");

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

int largest = num1, smallest = num1;

if (num2 > largest) {

largest = num2;

}

if (num3 > largest) {

largest = num3;

}

if (num2 < smallest) {

smallest = num2;

}

if (num3 < smallest) {

smallest = num3;

}

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

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

return 0;

}

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

#include <stdio.h>

int main() {

int num1, num2, num3;

printf("Enter three numbers: ");

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

int largest = num1, smallest = num1;

switch ((num2 > num1) + (num3 > num1) + (num2 > num3)) {

case 0:

largest = num1;

break;

case 1:

largest = (num2 > num3) ? num2 : num3;

break;

case 2:

largest = num2;

break;

case 3:

largest = num3;

break;

}

switch ((num2 < num1) + (num3 < num1) + (num2 < num3)) {

case 0:

smallest = num1;

break;

case 1:

smallest = (num2 < num3) ? num2 : num3;

break;

case 2:

smallest = num2;

break;

case 3:

smallest = num3;

break;

}

printf("Using switch-case:\n");

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

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

return 0;

}

3. Loops:

LAB EXERCISE 1: Prime Number Check

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

Ans.

#include <stdio.h>

int main() {

int num, i, isPrime = 1;

printf("Enter a number: ");

scanf("%d", &num);

if (num <= 1) {

isPrime = 0;

} else {

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

if (num % i == 0) {

isPrime = 0;

break;

}

}

}

if (isPrime) {

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

} else {

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

}

return 0;

}

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

Ans.

#include <stdio.h>

int main() {

int num, i, j, isPrime;

printf("Enter a number: ");

scanf("%d", &num);

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

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

isPrime = 1;

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

if (i % j == 0) {

isPrime = 0;

break;

}

}

if (isPrime) {

printf("%d ", i);

}

}

printf("\n");

return 0;

}

LAB EXERCISE 2: Multiplication Table

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

Ans.

#include <stdio.h>

int main() {

int num, i;

printf("Enter an integer: ");

scanf("%d", &num);

printf("Multiplication table of %d is:\n", num);

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

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

}

return 0;

}

• 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, i, range;

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

scanf("%d", &num);

printf("Enter the range for the multiplication table: ");

scanf("%d", &range);

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

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

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

}

return 0;

}

LAB EXERCISE 3: Sum of Digits

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

Ans.

#include <stdio.h>

int main() {

int num, sum = 0;

printf("Enter an integer: ");

scanf("%d", &num);

num = (num < 0) ? -num : num;

while (num > 0) {

sum += num % 10;

num /= 10;

}

printf("The sum of the digits is: %d\n", sum);

return 0;

}

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

Ans.

#include <stdio.h>

int main() {

int num, sum = 0, reversed = 0;

printf("Enter an integer: ");

scanf("%d", &num);

num = (num < 0) ? -num : num;

while (num > 0) {

sum += num % 10;

reversed = reversed \* 10 + num % 10;

num /= 10;

}

printf("The sum of the digits is: %d\n", sum);

printf("The reversed number is: %d\n", reversed);

return 0;

}

4. Arrays

LAB EXERCISE 1: Maximum and Minimum in Array

• 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.

Ans.

#include <stdio.h>

int main() {

int numbers[10];

int i;

int max, min;

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

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

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

}

max = min = numbers[0];

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

if(numbers[i] > max) {

max = numbers[i];

}

if(numbers[i] < min) {

min = numbers[i];

}

}

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

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

return 0;

}

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

Ans.

#include <stdio.h>

int main() {

int numbers[10];

int i, j, temp;

int max, min;

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

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

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

}

max = min = numbers[0];

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

if(numbers[i] > max) {

max = numbers[i];

}

if(numbers[i] < min) {

min = numbers[i];

}

}

for(i = 0; i < 9; i++) {

for(j = 0; j < 9 - i; j++) {

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

temp = numbers[j];

numbers[j] = numbers[j + 1];

numbers[j + 1] = temp;

}

}

}

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

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

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

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

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

}

printf("\n");

return 0;

}

LAB EXERCISE 2: Matrix Addition

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

Ans.

#include <stdio.h>

int main() {

int matrix1[2][2], matrix2[2][2], result[2][2];

int i, j;

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

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

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

scanf("%d", &matrix1[i][j]);

}

}

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

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

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

scanf("%d", &matrix2[i][j]);

}

}

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

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

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

printf("Resultant matrix after addition:\n");

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

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

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

}

printf("\n");

}

return 0;

}

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

Ans.

#include <stdio.h>

int main() {

int matrix1[3][3], matrix2[3][3], result[3][3];

int i, j, k;

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

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

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

scanf("%d", &matrix1[i][j]);

}

}

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

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

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

scanf("%d", &matrix2[i][j]);

}

}

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

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

result[i][j] = 0;

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

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

printf("Resultant matrix after multiplication:\n");

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

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

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

}

printf("\n");

}

return 0;

}

LAB EXERCISE 3: Sum of Array Elements

• 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.

Ans.

#include <stdio.h>

int main() {

int N, i, sum = 0;

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

scanf("%d", &N);

int numbers[N];

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

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

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

}

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

sum += numbers[i];

}

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

return 0;

}

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

Ans.

#include <stdio.h>

int main() {

int N, i, sum = 0;

float average;

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

scanf("%d", &N);

int numbers[N];

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

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

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

}

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

sum += numbers[i];

}

average = (float)sum / N;

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

printf("Average of all array elements: %.2f\n", average);

return 0;

}

5. Functions

LAB EXERCISE 1: Fibonacci Sequence

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

Ans.

#include <stdio.h>

int fibonacci(int n) {

if (n <= 1) {

return n;

} else {

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

}

}

int main() {

int N, i;

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

scanf("%d", &N);

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

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

printf("%d ", fibonacci(i));

}

printf("\n");

return 0;

}

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

Ans.

#include <stdio.h>

#include <time.h>

int fibonacci\_recursive(int n) {

if (n <= 1) {

return n;

} else {

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

}

}

int fibonacci\_iterative(int n) {

int a = 0, b = 1, c, i;

if (n == 0) return a;

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

c = a + b;

a = b;

b = c;

}

return b;

}

int main() {

int N;

clock\_t start\_time, end\_time;

double recursive\_time, iterative\_time;

printf("Enter the value of N (Nth Fibonacci number): ");

scanf("%d", &N);

start\_time = clock();

int result\_recursive = fibonacci\_recursive(N);

end\_time = clock();

recursive\_time = ((double) (end\_time - start\_time)) / CLOCKS\_PER\_SEC;

start\_time = clock();

int result\_iterative = fibonacci\_iterative(N);

end\_time = clock();

iterative\_time = ((double) (end\_time - start\_time)) / CLOCKS\_PER\_SEC;

printf("\nRecursive approach: Fibonacci(%d) = %d\n", N, result\_recursive);

printf("Time taken for recursive approach: %f seconds\n", recursive\_time);

printf("\nIterative approach: Fibonacci(%d) = %d\n", N, result\_iterative);

printf("Time taken for iterative approach: %f seconds\n", iterative\_time);

return 0;

}

LAB EXERCISE 2: Factorial Calculation

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

Ans.

#include <stdio.h>

long long factorial(int n) {

if (n <= 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (num < 0) {

printf("Factorial is not defined for negative numbers.\n");

} else {

printf("Factorial of %d is %lld\n", num, factorial(num));

}

return 0;

}

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

Ans.

#include <stdio.h>

#include <time.h>

long long factorial\_recursive(int n) {

if (n <= 1) {

return 1;

} else {

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

}

}

long long factorial\_iterative(int n) {

long long result = 1;

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

result \*= i;

}

return result;

}

int main() {

int num;

clock\_t start\_time, end\_time;

double recursive\_time, iterative\_time;

printf("Enter a number: ");

scanf("%d", &num);

if (num < 0) {

printf("Factorial is not defined for negative numbers.\n");

return 1;

}

start\_time = clock();

long long result\_recursive = factorial\_recursive(num);

end\_time = clock();

recursive\_time = ((double)(end\_time - start\_time)) / CLOCKS\_PER\_SEC;

start\_time = clock();

long long result\_iterative = factorial\_iterative(num);

end\_time = clock();

iterative\_time = ((double)(end\_time - start\_time)) / CLOCKS\_PER\_SEC;

printf("\nRecursive approach: Factorial of %d is %lld\n", num, result\_recursive);

printf("Time taken for recursive approach: %f seconds\n", recursive\_time);

printf("\nIterative approach: Factorial of %d is %lld\n", num, result\_iterative);

printf("Time taken for iterative approach: %f seconds\n", iterative\_time);

return 0;

}

LAB EXERCISE 3: Palindrome Check

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

Ans.

#include <stdio.h>

int is\_palindrome(int num) {

int original = num;

int reversed = 0, remainder;

while (num != 0) {

remainder = num % 10;

reversed = reversed \* 10 + remainder;

num /= 10;

}

return original == reversed;

}

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (is\_palindrome(num)) {

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

} else {

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

}

return 0;

}

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

Ans.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int is\_palindrome(char str[]) {

int start = 0;

int end = strlen(str) - 1;

while (start < end) {

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

return 0;

}

start++;

end--;

}

return 1;

}

int main() {

char str[100];

printf("Enter a string: ");

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

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

if (is\_palindrome(str)) {

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

} else {

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

}

return 0;

}

6. Strings

LAB EXERCISE 1: String Reversal

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

Ans.

#include <stdio.h>

#include <string.h>

void reverse\_string(char str[]) {

int start = 0;

int end = strlen(str) - 1;

char temp;

while (start < end) {

temp = str[start];

str[start] = str[end];

str[end] = temp;

start++;

end--;

}

}

int main() {

char str[100];

printf("Enter a string: ");

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

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

reverse\_string(str);

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

return 0;

}

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

Ans.

#include <stdio.h>

void reverse\_string(char str[]) {

int start = 0, end = 0;

char temp;

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

end++;

}

end--;

while (start < end) {

temp = str[start];

str[start] = str[end];

str[end] = temp;

start++;

end--;

}

}

int main() {

char str[100];

printf("Enter a string: ");

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

int i = 0;

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

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

str[i] = '\0';

break;

}

i++;

}

reverse\_string(str);

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

return 0;

}

LAB EXERCISE 2: Count Vowels and Consonants

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

Ans.

#include <stdio.h>

#include <ctype.h>

void count\_vowels\_and\_consonants(char str[], int \*vowel\_count, int \*consonant\_count) {

\*vowel\_count = 0;

\*consonant\_count = 0;

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

char ch = tolower(str[i]);

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

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

(\*vowel\_count)++;

} else {

(\*consonant\_count)++;

}

}

}

}

int main() {

char str[100];

int vowels, consonants;

printf("Enter a string: ");

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

count\_vowels\_and\_consonants(str, &vowels, &consonants);

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

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

return 0;

}

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

Ans.

#include <stdio.h>

#include <ctype.h>

void count\_chars(char str[], int \*vowel\_count, int \*consonant\_count, int \*digit\_count, int \*special\_count) {

\*vowel\_count = 0;

\*consonant\_count = 0;

\*digit\_count = 0;

\*special\_count = 0;

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

char ch = tolower(str[i]);

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

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

(\*vowel\_count)++;

} else {

(\*consonant\_count)++;

}

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

(\*digit\_count)++;

} else if (!isspace(str[i])) {

(\*special\_count)++;

}

}

}

int main() {

char str[100];

int vowels, consonants, digits, special;

printf("Enter a string: ");

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

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

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

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

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

printf("Special characters: %d\n", special);

return 0;

}

LAB EXERCISE 3: Word Count

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

Ans.

#include <stdio.h>

#include <ctype.h>

int count\_words(char str[]) {

int count = 0, in\_word = 0;

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

if (isspace(str[i])) {

in\_word = 0;

} else if (!in\_word) {

in\_word = 1;

count++;

}

}

return count;

}

int main() {

char str[100];

printf("Enter a sentence: ");

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

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

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

return 0;

}

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

Ans.

#include <stdio.h>

#include <ctype.h>

#include <string.h>

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

int max\_len = 0, current\_len = 0;

int start = 0;

int i = 0;

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

if (isspace(str[i]) || str[i+1] == '\0') {

if (current\_len > max\_len) {

max\_len = current\_len;

strncpy(longest, &str[start], current\_len);

longest[current\_len] = '\0';

}

current\_len = 0;

} else if (current\_len == 0) {

start = i;

}

current\_len++;

i++;

}

}

int count\_words(char str[]) {

int count = 0, in\_word = 0;

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

if (isspace(str[i])) {

in\_word = 0;

} else if (!in\_word) {

in\_word = 1;

count++;

}

}

return count;

}

int main() {

char str[200], longest[100];

printf("Enter a sentence: ");

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

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

find\_longest\_word(str, longest);

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

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

return 0;

}

Extra Logic Building Challenges

Lab Challenge 1: Armstrong Number

• 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)

#include <stdio.h>

#include <math.h>

int is\_armstrong(int num) {

int original\_num = num;

int sum = 0;

int digits = 0;

while (num != 0) {

digits++;

num /= 10;

}

num = original\_num;

while (num != 0) {

int digit = num % 10;

sum += pow(digit, digits);

num /= 10;

}

return sum == original\_num;

}

int main() {

int num;

printf("Enter a 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);

}

return 0;

}

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

Ans.

#include <stdio.h>

#include <math.h>

int is\_armstrong(int num) {

int original\_num = num;

int sum = 0;

int digits = 0;

while (num != 0) {

digits++;

num /= 10;

}

num = original\_num;

while (num != 0) {

int digit = num % 10;

sum += pow(digit, digits);

num /= 10;

}

return sum == original\_num;

}

int main() {

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

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

if (is\_armstrong(num)) {

printf("%d\n", num);

}

}

return 0;

}

Lab Challenge 2: Pascal’s Triangle

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

Ans.

#include <stdio.h>

int factorial(int n) {

int fact = 1;

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

fact \*= i;

}

return fact;

}

int combination(int n, int r) {

return factorial(n) / (factorial(r) \* factorial(n - r));

}

void print\_pascals\_triangle(int n) {

for (int row = 0; row < n; row++) {

for (int space = 0; space < n - row - 1; space++) {

printf(" ");

}

for (int col = 0; col <= row; col++) {

printf("%d ", combination(row, col));

}

printf("\n");

}

}

int main() {

int n;

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

scanf("%d", &n);

print\_pascals\_triangle(n);

return 0;

}

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

Ans.

#include <stdio.h>

int factorial(int n) {

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

return 1;

}

return n \* factorial(n - 1);

}

int combination(int n, int r) {

return factorial(n) / (factorial(r) \* factorial(n - r));

}

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

if (row >= n) {

return;

}

for (int space = 0; space < n - row - 1; space++) {

printf(" ");

}

for (int col = 0; col <= row; col++) {

printf("%d ", combination(row, col));

}

printf("\n");

print\_pascals\_triangle(n, row + 1); // Recursive call to print the next row

}

int main() {

int n;

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

scanf("%d", &n);

print\_pascals\_triangle(n, 0);

return 0;

}

Lab Challenge 3: Number Guessing Game

• 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.

Ans.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int target, guess, attempts = 10;

srand(time(0));

target = rand() % 100 + 1;

printf("Welcome to the number guessing game!\n");

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

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

while (attempts > 0) {

printf("Enter your guess: ");

scanf("%d", &guess);

if (guess > target) {

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

} else if (guess < target) {

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

} else {

printf("Congratulations! You guessed the number correctly.\n");

break;

}

attempts--;

if (attempts > 0) {

printf("You have %d attempts left.\n", attempts);

} else {

printf("Sorry, you've run out of attempts. The correct number was %d.\n", target);

}

}

return 0;

}

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

Ans.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int target, guess, attempts = 10;

srand(time(0));

target = rand() % 100 + 1;

printf("Welcome to the number guessing game!\n");

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

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

while (attempts > 0) {

printf("Enter your guess: ");

scanf("%d", &guess);

if (guess > target) {

printf("Too high! Try guessing a lower number.\n");

} else if (guess < target) {

printf("Too low! Try guessing a higher number.\n");

} else {

printf("Congratulations! You guessed the number correctly.\n");

break;

}

attempts--;

if (attempts > 0) {

printf("You have %d attempts left.\n", attempts);

} else {

printf("Sorry, you've run out of attempts. The correct number was %d.\n", target);

}

}

return 0;

}