---EXTRA LAB EXERCISES FOR IMPROVING PROGRAMMING LOGIC--\_

1. Operators:

LAB EXERCISE 1: Simple Calculator:

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.

Ans.

#include <stdio.h>

#include <stdlib.h>

int main()

{

float num1, num2, result;

char operator, ch;

do

{

printf("\n\n\t Enter any first numbers: ");

scanf("%f", &num1);

printf("\n\n\t Enter any second numbers: ");

scanf("%f", &num2);

printf("\n\n\t ----------MENU----------");

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

scanf(" %c", &operator);

switch (operator)

{

case '+':

result = num1 + num2;

printf("\n\n\t The result of Addition is: %.2f", result);

break;

case '-':

result = num1 - num2;

printf("\n\n\t The result of Subtraction is: %.2f", result);

break;

case '\*':

result = num1 \* num2;

printf("\n\n\t The result of Multiplication is: %.2f", result);

break;

Case '/':

if (num2 != 0)

{

result = num1 / num2;

printf("\n\n\t The result of Division is: %.2f", result);

} else

{

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

}

break;

default:

printf("\n\n\t Invalid operator! Please select a valid operator.");

break;

}

printf("\n\n\n\t Do you want to continue? (y/n) : ");

scanf(" %c",&ch);

}

while(ch=='y' || ch=='y');

}

LAB EXERCISE 2:

Check Number Properties:

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.

Ans.

#include <stdio.h>

main()

{

int num;

printf("\n\n\t Enter an integer: ");

scanf("%d", &num);

if (num % 2 == 0)

{

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

}

else

{

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

}

if (num > 0)

{

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

}

else if (num < 0)

{

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

}

else

{

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

}

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

{

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

}

else

{

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

}

}

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.

Ans.

#include <stdio.h>

main()

{

int marks;

printf("\n\n\t Enter the marks of the student: ");

scanf("%d", &marks);

if (marks <= 90)

{

printf("\n\n\t Grade A\n");

}

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

{

printf("\n\n\t Grade B\n");

}

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

{

printf("\n\n\t Grade C\n");

}

else if (marks <= 50)

{

printf("\n\n\t Grade D\n");

}

else

{

printf("\n\n\t Invalid input\n");

}

}

LAB EXERCISE 2:

Number Comparison:

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.

Ans.

#include <stdio.h>

main()

{

int num1, num2, num3;

printf("\n\n\t Enter first numbers: ");

scanf("%d", &num1);

printf("\n\n\t Enter second numbers: ");

scanf("%d", &num2);

printf("\n\n\t Enter three numbers: ");

scanf("%d", &num3);

// --- Finding the largest number using if-else ---

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

{

printf("\n\n\t The largest number (if-else statement) is: %d\n", num1);

}

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

{

printf("\n\n\t The largest number (if-else statement) is: %d\n", num2);

}

else

{

printf("\n\n\t The largest number (if-else statement) is: %d\n", num3);

}

// --- Finding the smallest number using if-else ---

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

{

printf("\n\n\t The smallest number (if-else statement) is: %d\n", num1);

}

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

{

printf("\n\n\t The smallest number (if-else statement) is: %d\n", num2);

}

else

{

printf("\n\n\t The smallest number (if-else statement) is: %d\n", num3);

}

// --- Finding the largest number using switch-case ---

int largest = 0;

switch (num1 >= num2 && num1 >= num3)

{

case 1: // num1 is the largest

largest = num1;

break;

default: // num1 is not the largest, check the others

switch (num2 >= num3)

{

case 1: // num2 is the largest

largest = num2;

break;

default: // num3 is the largest

largest = num3;

}

}

printf("\n\n\t The largest number (switch-case statement) is: %d\n", largest);

// --- Finding the smallest number using switch-case ---

int smallest = 0;

switch (num1 <= num2 && num1 <= num3)

{

case 1: // num1 is the smallest

smallest = num1;

break;

default: // num1 is not the smallest, check the others

switch (num2 <= num3)

{

case 1: // num2 is the smallest

smallest = num2;

break;

default: // num3 is the smallest

smallest = num3;

}

}

printf("\n\n\t The smallest number (switch-case statement) is: %d\n", smallest);

}

3. Loops:

LAB EXERCISE 1:

Prime Number Check:

Write a C program that checks whether a given number is a prime number using a for loop.

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

Ans.

#include <stdio.h>

main()

{

int num\_1, i;

printf("\n\n\t Enter a number: ");

scanf("%d", &num\_1);

for (i = 2; i <= num\_1 / 2; i++)

{

if (num\_1 % i == 0)

{

printf("\n\n\t %d is not a prime number.\n", num\_1);

}

}

printf("\n\n\t %d is a prime number.\n", num\_1);

{

int num, i, j, Prime;

printf("\n\n\t Prime numbers between 1 and %d are:\n", num\_1);

for (num = 2; num <=num\_1; num++)

{

Prime = 1;

for (i=2;i\*i<=num;i++)

{

if (num % i == 0)

{

Prime = 0;

break;

}

}

if (Prime)

{

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

}

}

printf("\n");

}

}

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.

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

Ans.

#include <stdio.h>

int main()

{

int num, i, range;

printf("\n\n\t Enter an integer number: ");

scanf("%d", &num);

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

scanf("%d", &range);

// Print the multiplication table

printf("\n\n\t Multiplication table for %d:\n", num);

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

{

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

}

}

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.

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

Ans.

#include<stdio.h>

main()

{

Int n, rem, sum=0;

printf("\n\n\t Enter any number to reverse : ");

scanf("%d",&n);

while(n>0)

{

rem=n%10; //1

printf("\t%d", rem); //7 3 1

sum=sum+rem;

n=n/10; //1

}

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

}

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.

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

Ans.

#include <stdio.h>

int main()

{

int arr[10], i, j, temp=0;

int max=0, min=0;

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

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

{

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

}

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];

}

}

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

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

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

{

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

{

if (arr[i] > arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

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

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

{

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

}

printf("\n");

}

LAB EXERCISE 2:

Matrix Addition:

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.

Ans.

#include<stdio.h>

Int main()

{

int mat1[3][3], mat2[3][3], mul[3][3];

int r, c, k;

printf("\n\n\n\t Input Matrix-1 ------------------\n\n");

for(r=0;r<3;r++)

{

for(c=0;c<3;c++)

{

printf("\n\n\t mat[%d][%d] : ", r, c);

scanf("%d",&mat1[r][c]);

}

}

printf("\n\n\n\t Input Matrix-2 ------------------\n\n");

for(r=0;r<3;r++)

{

for(c=0;c<3;c++)

{

printf("\n\n\t mat[%d][%d] : ", r, c);

scanf("%d",&mat2[r][c]);

}

}

printf("\n\n\n\t Matrix-1 ------------------\n\n");

for(r=0;r<3;r++)

{

for(c=0;c<3;c++)

{

printf(" %d", mat1[r][c]);

}

printf("\n");

}

printf("\n\n\n\t Matrix-2 ------------------\n\n");

for(r=0;r<3;r++)

{

for(c=0;c<3;c++)

{

printf(" %d", mat2[r][c]);

}

printf("\n");

}

printf("\n\n\n\t Multiplication of Matrix 1 & 2 ------------------\n\n");

for(r=0;r<3;r++)

{

for(c=0;c<3;c++)

{

mul[r][c]=0;

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

{

mul[r][c]=mul[r][c]+(mat1[r][k]\*mat2[k][c]);

//mul[r][c]+=mat1[r][k]\*mat2[k][c];

}

}

printf("\n");

}

for(r=0;r<3;r++)

{

for(c=0;c<3;c++)

{

printf(" %d", mul[r][c]);

}

printf("\n");

}

}

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.

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

Ans.

#include <stdio.h>

int main()

{

int n,sum = 0;

float average;

printf("\n\n\t Enter the number of elements: ");

scanf("%d", &n);

int numbers[n];

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

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

{

printf("\n\n\t Enter number %d: ", i + 1);

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

sum=sum+numbers[i];

}

average = (float)sum / n;

printf("\n\n\t Sum of the numbers: %d\n", sum);

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

}

5. Functions

LAB EXERCISE 1:

Fibonacci Sequence:

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 iterative and recursive methods. Compare their efficiency.

Ans.

#include <stdio.h>

#include <time.h>

// Recursive function

int fibonacci\_recursive(int n)

{

if (n <= 1)

{

return n;

}

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

}

// Iterative function

int fibonacci\_iterative(int n)

{

int a = 0, b = 1, c;

if (n <= 1)

{

return n;

}

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

{

c = a + b;

a = b;

b = c;

}

return b;

}

void print\_fibonacci\_sequence(int n)

{

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

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

{

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

}

printf("\n");

}

int main()

{

int N;

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

scanf("%d", &N);

print\_fibonacci\_sequence(N);

clock\_t start\_recursive = clock();

int result\_recursive = fibonacci\_recursive(N - 1);

clock\_t end\_recursive = clock();

double time\_recursive = ((double)(end\_recursive - start\_recursive))

/ CLOCKS\_PER\_SEC;

clock\_t start\_iterative = clock();

int result\_iterative = fibonacci\_iterative(N - 1);

clock\_t end\_iterative = clock();

double time\_iterative = ((double)(end\_iterative - start\_iterative))

/ CLOCKS\_PER\_SEC;

printf("\n\n\t Nth Fibonacci number using recursive method: %d\n", result\_recursive);

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

printf("\n\n\t Nth Fibonacci number using iterative method: %d\n", result\_iterative);

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

if (time\_recursive > time\_iterative)

{

printf("\n\n\t The iterative method is more efficient for large N.\n");

} else

{

printf("\n\n\t The recursive method is more efficient for small N.\n");

}

return 0;

}

LAB EXERCISE 2:

Factorial Calculation

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.

Ans.

#include <stdio.h>

#include <time.h>

// Iterative function

long long factorial\_iterative(int n)

{

long long result = 1;

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

{

result \*= i;

}

return result;

}

// Recursive function

long long factorial\_recursive(int n)

{

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

{

return 1;

} else

{

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

}

}

int main()

{

int num;

printf("\n\n\t Enter a number: ");

scanf("%d", &num);

// Measure time for iterative method

clock\_t start\_iterative = clock();

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

clock\_t end\_iterative = clock();

double time\_iterative = ((double)(end\_iterative - start\_iterative))

/ CLOCKS\_PER\_SEC;

// Measure time for recursive method

clock\_t start\_recursive = clock();

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

clock\_t end\_recursive = clock();

double time\_recursive = ((double)(end\_recursive - start\_recursive))

/ CLOCKS\_PER\_SEC;

// Output results

printf("\n\n\t Factorial of %d using iterative method: %lld\n",

num, result\_iterative);

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

printf("\n\n\t Factorial of %d using recursive method: %lld\n", num, result\_recursive);

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

if (time\_recursive > time\_iterative)

{

printf("\n\n\t The iterative method is more efficient for large numbers.\n");

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

{

printf("\n\n\t The recursive method is more efficient for small numbers.\n");

} else

{

printf("\n\n\t Both methods have the same efficiency for this input size.\n");

}

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.

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

Ans.

#include <stdio.h>

#include <string.h>

int is\_palindrome\_number(int num)

{

int original\_num = num;

int reversed\_num = 0;

int remainder;

while (num != 0)

{

remainder = num % 10;

reversed\_num = reversed\_num \* 10 + remainder;

num /= 10;

}

if (original\_num == reversed\_num)

{

return 1;

} else

{

return 0;

}

}

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

{

int start = 0;

int end = strlen(str) - 1;

while (start < end)

{

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

{

return 0;

}

start++;

end--;

}

return 1;

}

int main()

{

int num;

char str[100];

printf("\n\n\t 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);

}

printf("\n\n\t Enter a string: ");

scanf("%s", str);

if (is\_palindrome\_string(str))

{

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

} else

{

printf("\"%s\" is not a palindrome string.\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.

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

Ans.

#include <stdio.h>

// Function to reverse the string

void reverseString(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("\n\n\t Enter a string: ");

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

int i = 0;

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

{

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

{

str[i] = '\0';

break;

}

i++;

}

reverseString(str);

printf("\n\n\t 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.

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

Ans.

#include <stdio.h>

int main()

{

char str[100];

int vowels = 0, consonants = 0, digits = 0, specials = 0;

printf("\n\n\t Enter a string: ");

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

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

{

char ch = str[i];

switch (1)

{

case 1:

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

ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U')

{

vowels++;

break;

}

if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))

{

consonants++;

break;

}

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

{

digits++;

break;

}

specials++;

break;

}

}

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

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

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

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

return 0;

}

LAB EXERCISE 3: Word Count

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.

Ans.

#include<stdio.h>

main()

{

char str[10];

int i, count=0;

printf("\n\n\t Enter any string : ");

scanf("%s",&str);

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

{

printf(" %c", str[i]);

count++;

}

printf("\n\n\t Length of given string : %d", count);

}

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).

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

Ans.

#include <stdio.h>

#include <math.h>

int main()

{

int num, originalNum, remainder, sum, digits;

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

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

{

originalNum = num;

sum = 0;

digits = 0;

while (originalNum != 0)

{

originalNum /= 10;

digits++;

}

originalNum = num;

while (originalNum != 0)

{

remainder = originalNum % 10;

sum += pow(remainder, digits);

originalNum /= 10;

}

if (sum == 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.

Challenge: Implement the same program using a recursive function.

Ans.

#include <stdio.h>

void Triangle(int n)

{

for (int line = 0; line < n; line++)

{

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

{

printf(" ");

}

int num = 1;

for (int i = 0; i <= line; i++)

{

printf("%d ", num);

num = num \* (line - i) / (i + 1);

}

printf("\n");

}

}

int main()

{

int n;

printf("\n\n\t Enter the number of rows for Pascal's Triangle: ");

scanf("%d", &n);

Triangle(n);

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.

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 number, guess, attempts = 0, maxAttempts = 10;

srand(time(0));

number = rand() % 100 + 1;

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

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

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

while (attempts < maxAttempts)

{

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

scanf("%d", &guess);

attempts++;

if (guess < number)

{

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

} else if (guess > number)

{

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

} else

{

printf("\n\n\t Congratulations! You guessed the number %d

correctly in %d attempts.\n", number, attempts);

break;

}

}

if (guess != number)

{

printf("\n\n\t Sorry, you've used all your attempts. The correct number was %d.\n", number);

}

return 0;

}