ASSIGNMENT 1

///1. Write a C program to calculate factorial of a number.

#include <stdio.h>

// Function to calculate factorial

unsigned long long factorial(int n) {

    if (n == 0 || n == 1) {

        return 1;

    } else {

        return n \* factorial(n - 1);

    }

}

int main() {

    int num;

    // Input from user

    printf("Enter a non-negative integer: ");

    scanf("%d", &num);

    // Check if the number is non-negative

    if (num < 0) {

        printf("Please enter a non-negative integer.\n");

    } else {

        // Calculate and print factorial

        unsigned long long result = factorial(num);

        printf("Factorial of %d = %llu\n", num, result);

    }

    return 0;

}

///2. Write a C program to print all Prime numbers between 1 to N.

#include <stdio.h>

#include <stdbool.h>

// Function to check if a number is prime

bool isPrime(int num) {

    if (num <= 1) {

        return false;

    }

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

        if (num % i == 0) {

            return false;

        }

    }

    return true;

}

// Function to print prime numbers up to N

void printPrimes(int N) {

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

    for (int i = 2; i <= N; i++) {

        if (isPrime(i)) {

            printf("%d ", i);

        }

    }

    printf("\n");

}

int main() {

    int N;

    // Input from user

    printf("Enter a positive integer (N): ");

    scanf("%d", &N);

    // Check if the number is positive

    if (N <= 0) {

        printf("Please enter a positive integer.\n");

    } else {

        // Print prime numbers

        printPrimes(N);

    }

    return 0;

}

///3. Write a C program to find first and last digit of a number.

#include <stdio.h>

// Function to find the first digit of a number

int findFirstDigit(int num) {

    while (num >= 10) {

        num /= 10;

    }

    return num;

}

// Function to find the last digit of a number

int findLastDigit(int num) {

    return num % 10;

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Find and print the first and last digits

    int firstDigit = findFirstDigit(number);

    int lastDigit = findLastDigit(number);

    printf("First digit: %d\n", firstDigit);

    printf("Last digit: %d\n", lastDigit);

    return 0;

}

///4.Write a C program to swap first and last digits of a number.

#include <stdio.h>

// Function to count the number of digits in a number

int countDigits(int num) {

    int count = 0;

    while (num != 0) {

        num /= 10;

        count++;

    }

    return count;

}

// Function to swap the first and last digits of a number

int swapFirstAndLastDigits(int num) {

    int numDigits = countDigits(num);

    if (numDigits < 2) {

        // Nothing to swap if the number has 0 or 1 digit

        return num;

    }

    // Extract the first and last digits

    int firstDigit = num / (int)pow(10, numDigits - 1);

    int lastDigit = num % 10;

    // Remove the first and last digits from the number

    num %= (int)pow(10, numDigits - 1);

    num /= 10;

    // Construct the new number with swapped digits

    int result = lastDigit \* (int)pow(10, numDigits - 1) + num \* 10 + firstDigit;

    return result;

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Swap the first and last digits

    int result = swapFirstAndLastDigits(number);

    // Print the result

    printf("Number after swapping first and last digits: %d\n", result);

    return 0;

}

///5.Write a C program to find frequency of each digit in a given integer.

#include <stdio.h>

// Function to calculate the frequency of each digit

void digitFrequency(int num) {

    int count[10] = {0}; // Array to store the frequency of each digit

    // Count the frequency of each digit

    while (num != 0) {

        int digit = num % 10;

        count[digit]++;

        num /= 10;

    }

    // Print the frequency of each digit

    printf("Digit Frequency:\n");

    for (int i = 0; i < 10; i++) {

        if (count[i] > 0) {

            printf("Digit %d: %d times\n", i, count[i]);

        }

    }

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Calculate and print digit frequency

    digitFrequency(number);

    return 0;

}

ASSIGNMENT 2

///1. Write a C program to display the cube of the number upto n.

#include <stdio.h>

int main() {

    int n;

    // Input from user

    printf("Enter the value of n: ");

    scanf("%d", &n);

    // Display cubes of numbers up to n

    printf("Cubes of numbers up to %d are:\n", n);

    for (int i = 1; i <= n; i++) {

        printf("Cube of %d: %d\n", i, i \* i \* i);

    }

    return 0;

}

///2. Write a C program to display the N terms of odd numbers and their sum.

#include <stdio.h>

int main() {

    int n, i, sum = 0;

    // Input from user

    printf("Enter the value of n: ");

    scanf("%d", &n);

    // Display odd numbers and calculate their sum

    printf("First %d odd numbers are:\n", n);

    for (i = 1; i <= 2 \* n; i += 2) {

        printf("%d ", i);

        sum += i;

    }

    // Display the sum of odd numbers

    printf("\nSum of the first %d odd numbers: %d\n", n, sum);

    return 0;

}

///3. Write a C program to count number of digits in a number.

#include <stdio.h>

// Function to count the number of digits in a number

int countDigits(int num) {

    int count = 0;

    // Count digits using a loop

    while (num != 0) {

        num /= 10;

        count++;

    }

    return count;

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Count and print the number of digits

    int digitCount = countDigits(number);

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

    return 0;

}

///4. Write a C program to check whether a number is Armstrong number or not.

#include <stdio.h>

// Function to calculate the number of digits in a number

int countDigits(int num) {

    int count = 0;

    while (num != 0) {

        num /= 10;

        count++;

    }

    return count;

}

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

int isArmstrong(int num) {

    int originalNum = num;

    int numDigits = countDigits(num);

    int sum = 0;

    // Calculate the sum of digits raised to the power of the number of digits

    num = originalNum; // Reset num to the original value

    while (num > 0) {

        int digit = num % 10;

        // Calculate digit raised to the power of numDigits without using pow

        int power = 1;

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

            power \*= digit;

        }

        sum += power;

        num /= 10;

    }

    // Check if the sum is equal to the original number

    if (originalNum == sum) {

        return 1; // Armstrong number

    } else {

        return 0; // Not an Armstrong number

    }

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Check if the number is an Armstrong number

    if (isArmstrong(number)) {

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

    } else {

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

    }

    return 0;

}

///5. Write a C program to check whether a number is Palindrome or not.

#include <stdio.h>

// Function to check if a number is a palindrome

int isPalindrome(int num) {

    int originalNum = num;

    int reversedNum = 0;

    // Reverse the digits of the number

    while (num > 0) {

        int digit = num % 10;

        reversedNum = reversedNum \* 10 + digit;

        num /= 10;

    }

    // Check if the reversed number is equal to the original number

    if (originalNum == reversedNum) {

        return 1; // Palindrome

    } else {

        return 0; // Not a palindrome

    }

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Check if the number is a palindrome

    if (isPalindrome(number)) {

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

    } else {

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

    }

    return 0;

}

ASSIGNMENT 3 & 4

///1. Write a C program to find maximum between two numbers.

#include <stdio.h>

int main() {

    int num1, num2;

    printf("Enter two numbers: ");

    scanf("%d %d", &num1, &num2);

    if (num1 > num2)

        printf("Maximum is %d\n", num1);

    else

        printf("Maximum is %d\n", num2);

    return 0;

}

///2. Write a C program to find maximum between three numbers.

#include <stdio.h>

int main() {

    int num1, num2, num3;

    printf("Enter three numbers: ");

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

    if (num1 >= num2 && num1 >= num3)

        printf("Maximum is %d\n", num1);

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

        printf("Maximum is %d\n", num2);

    else

        printf("Maximum is %d\n", num3);

    return 0;

}

///3. Write a C program to check whether a number is negative, positive or zero.

#include <stdio.h>

int main() {

    int num;

    printf("Enter a number: ");

    scanf("%d", &num);

    if (num > 0)

        printf("Positive\n");

    else if (num < 0)

        printf("Negative\n");

    else

        printf("Zero\n");

    return 0;

}

///4. Write a C program to check whether a number is divisible by 5 and 11 or not.

#include <stdio.h>

int main() {

    int num;

    printf("Enter a number: ");

    scanf("%d", &num);

    if (num % 5 == 0 && num % 11 == 0)

        printf("Divisible by both 5 and 11\n");

    else if (num % 5 == 0)

        printf("Divisible by 5 but not by 11\n");

    else if (num % 11 == 0)

        printf("Divisible by 11 but not by 5\n");

    else

        printf("Not divisible by 5 and 11\n");

    return 0;

}

///5. Write a C program to check whether a number is even or odd.

#include <stdio.h>

int main() {

    int num;

    printf("Enter a number: ");

    scanf("%d", &num);

    if (num % 2 == 0)

        printf("Even\n");

    else

        printf("Odd\n");

    return 0;

}

///6. Write a C program to check whether a year is leap year or not.

#include <stdio.h>

int main() {

    int year;

    printf("Enter a year: ");

    scanf("%d", &year);

    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))

        printf("Leap year\n");

    else

        printf("Not a leap year\n");

    return 0;

}

///7. Write a C program to check whether a character is alphabet or not.

#include <stdio.h>

#include <ctype.h>

int main() {

    char word[100];  // Assuming a maximum word length of 100 characters

    printf("Enter a word: ");

    scanf("%s", word);

    int isAlphabet = 1;  // Assume the word is an alphabet until proven otherwise

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

        if (!isalpha(word[i])) {

            isAlphabet = 0;  // Set to 0 if any non-alphabetic character is found

            break;

        }

    }

    if (isAlphabet) {

        printf("The entire word is composed of alphabetic characters\n");

    } else {

        printf("The word contains non-alphabetic characters\n");

    }

    return 0;

}

///8. Write a C program to input any alphabet and check whether it is vowel or consonant.

#include <stdio.h>

int main() {

    char ch;

    printf("Enter an alphabet: ");

    scanf(" %c", &ch);

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

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

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

            printf("Vowel\n");

        else

            printf("Consonant\n");

    } else {

        printf("Not an alphabet\n");

    }

    return 0;

}

///9. Write a C program to input any character and check whether it is alphabet, digit or special character.

#include <stdio.h>

#include <ctype.h>

int main() {

    char word[100];  // Assuming a maximum word length of 100 characters

    printf("Enter a word: ");

    scanf("%s", word);

    int hasAlphabet = 0;

    int hasDigit = 0;

    int hasSpecialChar = 0;

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

        if (isalpha(word[i])) {

            hasAlphabet = 1;

        } else if (isdigit(word[i])) {

            hasDigit = 1;

        } else {

            hasSpecialChar = 1;

        }

    }

    if (hasAlphabet && hasDigit && hasSpecialChar) {

        printf("The word contains a combination of alphabetic characters, digits, and special characters\n");

    } else if (hasAlphabet && hasDigit) {

        printf("The word contains a combination of alphabetic characters and digits\n");

    } else if (hasAlphabet && hasSpecialChar) {

        printf("The word contains a combination of alphabetic characters and special characters\n");

    } else if (hasDigit && hasSpecialChar) {

        printf("The word contains a combination of digits and special characters\n");

    } else if (hasAlphabet) {

        printf("The word contains only alphabetic characters\n");

    } else if (hasDigit) {

        printf("The word contains only digits\n");

    } else if (hasSpecialChar) {

        printf("The word contains only special characters\n");

    } else {

        printf("The word is empty\n");

    }

    return 0;

}

ASSIGNMENT 5 & 6

///Write a C program to find power of a number using for loop.

#include <stdio.h>

// Function to calculate the power of a number using a for loop

long long power(int base, int exponent) {

    long long result = 1;

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

        result \*= base;

    }

    return result;

}

int main() {

    int base, exponent;

    // Input from user

    printf("Enter the base: ");

    scanf("%d", &base);

    printf("Enter the exponent: ");

    scanf("%d", &exponent);

    // Calculate and print the power

    long long result = power(base, exponent);

    printf("%d raised to the power %d is: %lld\n", base, exponent, result);

    return 0;

}

///2. Write a C program to find all factors of a number.

#include <stdio.h>

// Function to find and print all factors of a number

void findFactors(int num) {

    printf("Factors of %d are: ", num);

    for (int i = 1; i <= num; i++) {

        if (num % i == 0) {

            printf("%d ", i);

        }

    }

    printf("\n");

}

int main() {

    int number;

    // Input from user

    printf("Enter a positive integer: ");

    scanf("%d", &number);

    // Check if the number is positive

    if (number <= 0) {

        printf("Please enter a positive integer.\n");

        return 1;

    }

    // Find and print factors

    findFactors(number);

    return 0;

}

///3. Write a C program to calculate factorial of a number.

#include <stdio.h>

// Function to calculate the factorial of a number

long long calculateFactorial(int num) {

    if (num == 0 || num == 1) {

        return 1;

    } else {

        return num \* calculateFactorial(num - 1);

    }

}

int main() {

    int number;

    // Input from user

    printf("Enter a non-negative integer: ");

    scanf("%d", &number);

    // Check if the number is non-negative

    if (number < 0) {

        printf("Please enter a non-negative integer.\n");

        return 1;

    }

    // Calculate and print the factorial

    long long factorial = calculateFactorial(number);

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

    return 0;

}

///4. Write a C program to find HCF (GCD) of two numbers.

#include <stdio.h>

// Function to find the Greatest Common Divisor (GCD) using Euclidean Algorithm

int findGCD(int a, int b) {

    while (b != 0) {

        int temp = b;

        b = a % b;

        a = temp;

    }

    return a;

}

int main() {

    int num1, num2;

    // Input from user

    printf("Enter the first number: ");

    scanf("%d", &num1);

    printf("Enter the second number: ");

    scanf("%d", &num2);

    // Find and print the GCD (HCF)

    int gcd = findGCD(num1, num2);

    printf("GCD (HCF) of %d and %d is: %d\n", num1, num2, gcd);

    return 0;

}

///5. Write a C program to find LCM of two numbers.

#include <stdio.h>

// Function to find the Greatest Common Divisor (GCD) using Euclidean Algorithm

int findGCD(int a, int b) {

    while (b != 0) {

        int temp = b;

        b = a % b;

        a = temp;

    }

    return a;

}

// Function to find the Least Common Multiple (LCM) of two numbers

int findLCM(int a, int b) {

    // LCM \* GCD = a \* b

    // Therefore, LCM = (a \* b) / GCD

    int gcd = findGCD(a, b);

    int lcm = (a \* b) / gcd;

    return lcm;

}

int main() {

    int num1, num2;

    // Input from user

    printf("Enter the first number: ");

    scanf("%d", &num1);

    printf("Enter the second number: ");

    scanf("%d", &num2);

    // Find and print the LCM

    int lcm = findLCM(num1, num2);

    printf("LCM of %d and %d is: %d\n", num1, num2, lcm);

    return 0;

}

///6. Write a C program to check whether a number is Prime number or not.

#include <stdio.h>

#include <stdbool.h>

// Function to check if a number is prime

bool isPrime(int num) {

    if (num <= 1) {

        return false;

    }

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

        if (num % i == 0) {

            return false;

        }

    }

    return true;

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Check if the number is prime

    if (isPrime(number)) {

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

    } else {

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

    }

    return 0;

}

///7. Write a C program to print all Prime numbers between 1 to n.

#include <stdio.h>

#include <stdbool.h>

// Function to check if a number is prime

bool isPrime(int num) {

    if (num <= 1) {

        return false;

    }

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

        if (num % i == 0) {

            return false;

        }

    }

    return true;

}

// Function to print all prime numbers between 1 and n

void printPrimes(int n) {

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

    for (int i = 2; i <= n; i++) {

        if (isPrime(i)) {

            printf("%d\n", i);

        }

    }

}

int main() {

    int number;

    // Input from user

    printf("Enter a positive integer (n): ");

    scanf("%d", &number);

    // Check if the number is positive

    if (number <= 0) {

        printf("Please enter a positive integer.\n");

        return 1;

    }

    // Print prime numbers

    printPrimes(number);

    return 0;

}

///8. Write a C program to find sum of all prime numbers between 1 to n.

#include <stdio.h>

#include <stdbool.h>

// Function to check if a number is prime

bool isPrime(int num) {

    if (num <= 1) {

        return false;

    }

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

        if (num % i == 0) {

            return false;

        }

    }

    return true;

}

// Function to find the sum of prime numbers between 1 and n

int sumOfPrimes(int n) {

    int sum = 0;

    for (int i = 2; i <= n; i++) {

        if (isPrime(i)) {

            sum += i;

        }

    }

    return sum;

}

int main() {

    int number;

    // Input from user

    printf("Enter a positive integer (n): ");

    scanf("%d", &number);

    // Check if the number is positive

    if (number <= 0) {

        printf("Please enter a positive integer.\n");

        return 1;

    }

    // Calculate and print the sum of prime numbers

    int result = sumOfPrimes(number);

    printf("Sum of prime numbers between 1 and %d: %d\n", number, result);

    return 0;

}

///9. Write a C program to find all prime factors of a number.

#include <stdio.h>

// Function to print all prime factors of a number

void primeFactors(int num) {

    // Print the number of 2s that divide num

    while (num % 2 == 0) {

        printf("%d ", 2);

        num = num / 2;

    }

    // Consider odd factors starting from 3

    for (int i = 3; i \* i <= num; i = i + 2) {

        // Print i as a factor while i divides num

        while (num % i == 0) {

            printf("%d ", i);

            num = num / i;

        }

    }

    // If num becomes a prime number greater than 2

    if (num > 2) {

        printf("%d ", num);

    }

}

int main() {

    int number;

    // Input from user

    printf("Enter a positive integer: ");

    scanf("%d", &number);

    // Check if the number is positive

    if (number <= 0) {

        printf("Please enter a positive integer.\n");

        return 1;

    }

    // Print prime factors

    printf("Prime factors of %d are: ", number);

    primeFactors(number);

    return 0;

}

///10. Write a C program to check whether a number is Armstrong number or not.

#include <stdio.h>

// Function to calculate the number of digits in a number

int countDigits(int num) {

    int count = 0;

    while (num != 0) {

        num /= 10;

        count++;

    }

    return count;

}

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

int isArmstrong(int num) {

    int originalNum = num;

    int numDigits = countDigits(num);

    int sum = 0;

    // Calculate the sum of digits raised to the power of the number of digits

    while (num > 0) {

        int digit = num % 10;

        // Use integer multiplication instead of pow

        int digitPower = 1;

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

            digitPower \*= digit;

        }

        sum += digitPower;

        num /= 10;

    }

    // Check if the sum is equal to the original number

    return (originalNum == sum);

}

int main() {

    int number;

    // Input from user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Check if the number is an Armstrong number

    if (isArmstrong(number)) {

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

    } else {

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

    }

    return 0;

}

///11. Write a C program to print all Armstrong numbers between 1 to n.

#include <stdio.h>

#include <stdio.h>

#include <math.h>

// Function to calculate the number of digits in a number

int countDigits(int num) {

    int count = 0;

    while (num != 0) {

        num /= 10;

        count++;

    }

    return count;

}

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

int isArmstrong(int num) {

    int originalNum = num;

    int numDigits = countDigits(num);

    int sum = 0;

    // Calculate the sum of digits raised to the power of the number of digits

    while (num > 0) {

        int digit = num % 10;

        // Use integer multiplication instead of pow

        int digitPower = 1;

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

            digitPower \*= digit;

        }

        sum += digitPower;

        num /= 10;

    }

    // Check if the sum is equal to the original number

    return (originalNum == sum);

}

int main() {

    int n;

    // Input from user

    printf("Enter a positive integer (n): ");

    scanf("%d", &n);

    // Check if the number is positive

    if (n <= 0) {

        printf("Please enter a positive integer.\n");

        return 1;

    }

    // Print Armstrong numbers between 1 and n

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

    for (int i = 1; i <= n; i++) {

        if (isArmstrong(i)) {

            printf("%d\n", i);

        }

    }

    return 0;

}

///Write a C program to print the following pyramid

///\*\*\*\*\*

///\*\*\*\*

///\*\*\*

///\*\*

///\*

#include <stdio.h>

int main() {

    int rows;

    // Input from user

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    // Print the pyramid pattern

    for (int i = rows; i >= 1; i--) {

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

            printf("\*");

        }

        printf("\n");

    }

    return 0;

}

///Write a C program to print the following pyramid

///     \*

///    \* \*

///   \* \* \*

///  \* \* \* \*

/// \* \* \* \* \*

#include <stdio.h>

int main() {

    int rows;

    // Input from user

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    // Print the pyramid pattern

    for (int i = 1; i <= rows; i++) {

        // Print spaces

        for (int j = 1; j <= rows - i; j++) {

            printf(" ");

        }

        // Print stars

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

            printf("\* ");

        }

        printf("\n");

    }

    return 0;

}

///Write a c program to draw the following pyramid.

///                0

///             0  1

///          0  1  0

///       0  1  0  1

///    0  1  0  1  0

#include <stdio.h>

int main() {

    int rows;

    // Input from user

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    // Draw the pyramid

    for (int i = 0; i < rows; i++) {

        // Print leading spaces

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

            printf("   ");

        }

        // Print 0 and 1 alternatively

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

            if (k % 2 == 0) {

                printf("0  ");

            } else {

                printf("1  ");

            }

        }

        // Move to the next line

        printf("\n");

    }

    return 0;

}

///Write a C program to print the following pyramid

///              1

///            1 2 3

///          1 2 3 4 5

///        1 2 3 4 5 6 7

///      1 2 3 4 5 6 7 8 9

#include <stdio.h>

int main() {

    int rows;

    // Input from user

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    // Print the pyramid

    for (int i = 1; i <= rows; i++) {

        // Print leading spaces

        for (int j = 1; j <= rows - i; j++) {

            printf("  ");

        }

        // Print numbers

        for (int k = 1; k <= 2 \* i - 1; k++) {

            printf("%d ", k);

        }

        // Move to the next line

        printf("\n");

    }

    return 0;

}

ASSIGNMENT 7

///1. Write a program to store marks for n number of student in an array and print their marks.

#include <stdio.h>

int main() {

    int n;

    // Input the number of students

    printf("Enter the number of students: ");

    scanf("%d", &n);

    // Check if the number of students is valid

    if (n <= 0) {

        printf("Please enter a valid number of students.\n");

        return 1;

    }

    // Declare an array to store marks for n students

    int marks[n];

    // Input marks for each student

    for (int i = 0; i < n; i++) {

        printf("Enter marks for student %d: ", i + 1);

        scanf("%d", &marks[i]);

    }

    // Print the entered marks

    printf("\nMarks for %d students:\n", n);

    for (int i = 0; i < n; i++) {

        printf("Student %d: %d\n", i + 1, marks[i]);

    }

    return 0;

}

///2. Write a program which stores the marks of subject Mathematics and English of n number of students in an array and then prints their individual total marks.

#include <stdio.h>

int main() {

    int n;

    // Input the number of students

    printf("Enter the number of students: ");

    scanf("%d", &n);

    // Check if the number of students is valid

    if (n <= 0) {

        printf("Please enter a valid number of students.\n");

        return 1;

    }

    // Declare a 2D array to store marks for Mathematics and English for n students

    int marks[n][2];  // Assuming 2 subjects: Mathematics and English

    // Input marks for each student in Mathematics and English

    for (int i = 0; i < n; i++) {

        printf("Enter marks for Mathematics for student %d: ", i + 1);

        scanf("%d", &marks[i][0]);

        printf("Enter marks for English for student %d: ", i + 1);

        scanf("%d", &marks[i][1]);

    }

    // Calculate and print the total marks for each student

    printf("\nTotal Marks for %d students:\n", n);

    for (int i = 0; i < n; i++) {

        int totalMarks = marks[i][0] + marks[i][1];

        printf("Student %d: Total Marks = %d\n", i + 1, totalMarks);

    }

    return 0;

}

///3. Write a program to insert an element in an array in a particular position.

#include <stdio.h>

// Function to display the elements of an array

void displayArray(int arr[], int size) {

    printf("Array elements: ");

    for (int i = 0; i < size; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

// Function to insert an element at a specified position in an array

void insertElement(int arr[], int \*size, int element, int position) {

    // Check if the position is valid

    if (position < 0 || position > \*size) {

        printf("Invalid position for insertion.\n");

        return;

    }

    // Shift elements to the right to make space for the new element

    for (int i = \*size; i > position; i--) {

        arr[i] = arr[i - 1];

    }

    // Insert the new element at the specified position

    arr[position] = element;

    // Increment the size of the array

    (\*size)++;

}

int main() {

    int size, element, position;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    // Check if the size is valid

    if (size <= 0) {

        printf("Please enter a valid size for the array.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[size];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", size);

    for (int i = 0; i < size; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Display the original array

    printf("\nOriginal ");

    displayArray(arr, size);

    // Input the element to be inserted and its position

    printf("\nEnter the element to be inserted: ");

    scanf("%d", &element);

    printf("Enter the position to insert the element: ");

    scanf("%d", &position);

    // Insert the element at the specified position

    insertElement(arr, &size, element, position);

    // Display the array after insertion

    printf("\nArray after insertion ");

    displayArray(arr, size);

    return 0;

}

///4. Write a program to delete an element from a particular position of an array.

#include <stdio.h>

// Function to display the elements of an array

void displayArray(int arr[], int size) {

    printf("Array elements: ");

    for (int i = 0; i < size; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

// Function to delete an element at a specified position in an array

void deleteElement(int arr[], int \*size, int position) {

    // Check if the position is valid

    if (position < 0 || position >= \*size) {

        printf("Invalid position for deletion.\n");

        return;

    }

    // Shift elements to the left to fill the gap left by the deleted element

    for (int i = position; i < \*size - 1; i++) {

        arr[i] = arr[i + 1];

    }

    // Decrement the size of the array

    (\*size)--;

}

int main() {

    int size, position;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    // Check if the size is valid

    if (size <= 0) {

        printf("Please enter a valid size for the array.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[size];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", size);

    for (int i = 0; i < size; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Display the original array

    printf("\nOriginal ");

    displayArray(arr, size);

    // Input the position to delete the element

    printf("\nEnter the position to delete the element: ");

    scanf("%d", &position);

    // Delete the element at the specified position

    deleteElement(arr, &size, position);

    // Display the array after deletion

    printf("\nArray after deletion ");

    displayArray(arr, size);

    return 0;

}

///5. Write a program to convert a decimal number taken as input from user to corresponding binary number and store the result in an array.

#include <stdio.h>

// Function to convert decimal to binary and store in an array

void decimalToBinary(int decimal, int binaryArray[], int \*size) {

    \*size = 0;

    // Convert decimal to binary

    while (decimal > 0) {

        binaryArray[\*size] = decimal % 2;

        decimal /= 2;

        (\*size)++;

    }

    // If the input decimal is 0

    if (\*size == 0) {

        binaryArray[\*size] = 0;

        (\*size)++;

    }

}

// Function to display the elements of an array

void displayArray(int arr[], int size) {

    printf("Binary representation: ");

    for (int i = size - 1; i >= 0; i--) {

        printf("%d", arr[i]);

    }

    printf("\n");

}

int main() {

    int decimal, size;

    // Input the decimal number

    printf("Enter a decimal number: ");

    scanf("%d", &decimal);

    // Check if the decimal number is non-negative

    if (decimal < 0) {

        printf("Please enter a non-negative decimal number.\n");

        return 1;

    }

    // Declare an array to store the binary representation

    int binaryArray[32];  // Assuming a 32-bit binary representation

    // Convert decimal to binary and store in the array

    decimalToBinary(decimal, binaryArray, &size);

    // Display the binary representation

    displayArray(binaryArray, size);

    return 0;

}

///6. Write a program to input a binary number in an array and convert into corresponding decimal number.

#include <stdio.h>

#include <math.h>

// Function to convert binary to decimal

int binaryToDecimal(long long binary) {

    int decimal = 0, i = 0;

    // Convert binary to decimal

    while (binary != 0) {

        int remainder = binary % 10;

        decimal += remainder \* pow(2, i);

        binary /= 10;

        i++;

    }

    return decimal;

}

int main() {

    long long binary;

    // Input the binary number

    printf("Enter a binary number: ");

    scanf("%lld", &binary);

    // Check if the binary number is non-negative

    if (binary < 0) {

        printf("Please enter a non-negative binary number.\n");

        return 1;

    }

    // Convert binary to decimal

    int decimal = binaryToDecimal(binary);

    // Display the decimal equivalent

    printf("Decimal equivalent: %d\n", decimal);

    return 0;

}

///7. Write a program to find the smallest and the largest elements in an array.

#include <stdio.h>

// Function to find the smallest and largest elements in an array

void findSmallestAndLargest(int arr[], int size, int \*smallest, int \*largest) {

    // Assume the first element as the initial smallest and largest

    \*smallest = \*largest = arr[0];

    for (int i = 1; i < size; i++) {

        if (arr[i] < \*smallest) {

            \*smallest = arr[i];

        } else if (arr[i] > \*largest) {

            \*largest = arr[i];

        }

    }

}

int main() {

    int size;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    // Check if the size is valid

    if (size <= 0) {

        printf("Please enter a valid size for the array.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[size];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", size);

    for (int i = 0; i < size; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Find the smallest and largest elements in the array

    int smallest, largest;

    findSmallestAndLargest(arr, size, &smallest, &largest);

    // Display the result

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

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

    return 0;

}

///8. Write a program for deleting duplicate elements in an array.

#include <stdio.h>

// Function to remove duplicate elements from an array

int removeDuplicates(int arr[], int size) {

    int newSize = 1;

    for (int i = 1; i < size; i++) {

        int isDuplicate = 0;

        // Check if the current element is a duplicate

        for (int j = 0; j < newSize; j++) {

            if (arr[i] == arr[j]) {

                isDuplicate = 1;

                break;

            }

        }

        // If not a duplicate, add it to the array

        if (!isDuplicate) {

            arr[newSize] = arr[i];

            newSize++;

        }

    }

    return newSize;

}

int main() {

    int size;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    // Check if the size is valid

    if (size <= 0) {

        printf("Please enter a valid size for the array.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[size];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", size);

    for (int i = 0; i < size; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Remove duplicates from the array

    int newSize = removeDuplicates(arr, size);

    // Display the array after removing duplicates

    printf("\nArray after removing duplicates:\n");

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

        printf("%d ", arr[i]);

    }

    printf("\n");

    return 0;

}

///9. Write a program to search a particular element in an array.

#include <stdio.h>

// Function to search for an element in an array

int searchElement(int arr[], int size, int target) {

    for (int i = 0; i < size; i++) {

        if (arr[i] == target) {

            return i;  // Return the index if the element is found

        }

    }

    return -1;  // Return -1 if the element is not found

}

int main() {

    int size, target;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    // Check if the size is valid

    if (size <= 0) {

        printf("Please enter a valid size for the array.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[size];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", size);

    for (int i = 0; i < size; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Input the element to search

    printf("Enter the element to search: ");

    scanf("%d", &target);

    // Search for the element in the array

    int index = searchElement(arr, size, target);

    // Display the result

    if (index != -1) {

        printf("Element %d found at index %d.\n", target, index);

    } else {

        printf("Element %d not found in the array.\n", target);

    }

    return 0;

}

///10. Write a program to sort n elements (ascending order).

#include <stdio.h>

// Function to perform bubble sort

void bubbleSort(int arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

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

            // Swap if the element found is greater than the next element

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

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

}

// Function to display the elements of an array

void displayArray(int arr[], int n) {

    printf("Sorted array: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main() {

    int n;

    // Input the number of elements

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

    scanf("%d", &n);

    // Check if the number of elements is valid

    if (n <= 0) {

        printf("Please enter a valid number of elements.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[n];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", n);

    for (int i = 0; i < n; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Perform bubble sort

    bubbleSort(arr, n);

    // Display the sorted array

    displayArray(arr, n);

    return 0;

}

///11. Write a program to find second highest number from the array without using sorting.

#include <stdio.h>

// Function to find the second highest number in an array

int findSecondHighest(int arr[], int n) {

    // Check if the array has at least two elements

    if (n < 2) {

        printf("Please enter at least two elements for finding the second highest.\n");

        return -1; // Return a special value indicating an error

    }

    int firstHighest, secondHighest;

    // Initialize firstHighest and secondHighest based on the first two elements

    if (arr[0] > arr[1]) {

        firstHighest = arr[0];

        secondHighest = arr[1];

    } else {

        firstHighest = arr[1];

        secondHighest = arr[0];

    }

    // Iterate through the array to find the second highest

    for (int i = 2; i < n; i++) {

        if (arr[i] > firstHighest) {

            secondHighest = firstHighest;

            firstHighest = arr[i];

        } else if (arr[i] > secondHighest && arr[i] != firstHighest) {

            secondHighest = arr[i];

        }

    }

    return secondHighest;

}

int main() {

    int n;

    // Input the number of elements

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

    scanf("%d", &n);

    // Check if the number of elements is valid

    if (n <= 0) {

        printf("Please enter a valid number of elements.\n");

        return 1;

    }

    // Declare an array of the specified size

    int arr[n];

    // Input elements of the array

    printf("Enter %d elements for the array:\n", n);

    for (int i = 0; i < n; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Find the second highest number

    int secondHighest = findSecondHighest(arr, n);

    // Display the result

    if (secondHighest != -1) {

        printf("Second highest number: %d\n", secondHighest);

    }

    return 0;

}

///12. Write a program to perform addition and subtraction between two matrices.

#include <stdio.h>

// Function to input elements of a matrix

void inputMatrix(int matrix[][100], int rows, int columns) {

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

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

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

        }

    }

}

// Function to display a matrix

void displayMatrix(int matrix[][100], int rows, int columns) {

    printf("Matrix:\n");

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

        printf("\n");

    }

}

// Function to add two matrices

void addMatrices(int matrix1[][100], int matrix2[][100], int result[][100], int rows, int columns) {

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

    }

}

// Function to subtract two matrices

void subtractMatrices(int matrix1[][100], int matrix2[][100], int result[][100], int rows, int columns) {

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

    }

}

int main() {

    int rows, columns;

    // Input the number of rows and columns for the matrices

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    printf("Enter the number of columns: ");

    scanf("%d", &columns);

    // Check if the matrix dimensions are valid

    if (rows <= 0 || columns <= 0) {

        printf("Please enter valid dimensions for the matrices.\n");

        return 1;

    }

    // Declare two matrices and the result matrix

    int matrix1[100][100], matrix2[100][100], result[100][100];

    // Input elements for the first matrix

    printf("\nFor the first matrix:\n");

    inputMatrix(matrix1, rows, columns);

    // Input elements for the second matrix

    printf("\nFor the second matrix:\n");

    inputMatrix(matrix2, rows, columns);

    // Perform addition

    addMatrices(matrix1, matrix2, result, rows, columns);

    // Display the result of addition

    printf("\nAddition Result:\n");

    displayMatrix(result, rows, columns);

    // Perform subtraction

    subtractMatrices(matrix1, matrix2, result, rows, columns);

    // Display the result of subtraction

    printf("\nSubtraction Result:\n");

    displayMatrix(result, rows, columns);

    return 0;

}

///13. Write a program to transpose a matrix.

#include <stdio.h>

// Function to input elements of a matrix

void inputMatrix(int matrix[][100], int rows, int columns) {

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

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

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

        }

    }

}

// Function to display a matrix

void displayMatrix(int matrix[][100], int rows, int columns) {

    printf("Matrix:\n");

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

        printf("\n");

    }

}

// Function to transpose a matrix

void transposeMatrix(int matrix[][100], int transposedMatrix[][100], int rows, int columns) {

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

            transposedMatrix[j][i] = matrix[i][j];

        }

    }

}

int main() {

    int rows, columns;

    // Input the number of rows and columns for the matrix

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    printf("Enter the number of columns: ");

    scanf("%d", &columns);

    // Check if the matrix dimensions are valid

    if (rows <= 0 || columns <= 0) {

        printf("Please enter valid dimensions for the matrix.\n");

        return 1;

    }

    // Declare a matrix and the transposed matrix

    int matrix[100][100], transposedMatrix[100][100];

    // Input elements for the matrix

    printf("\nFor the matrix:\n");

    inputMatrix(matrix, rows, columns);

    // Transpose the matrix

    transposeMatrix(matrix, transposedMatrix, rows, columns);

    // Display the original matrix

    printf("\nOriginal Matrix:\n");

    displayMatrix(matrix, rows, columns);

    // Display the transposed matrix

    printf("\nTransposed Matrix:\n");

    displayMatrix(transposedMatrix, columns, rows);

    return 0;

}

///14. Write a program to add the elements of each row and each column of a matrix.

#include <stdio.h>

// Function to input elements of a matrix

void inputMatrix(int matrix[][100], int rows, int columns) {

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

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

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

        }

    }

}

// Function to display a matrix

void displayMatrix(int matrix[][100], int rows, int columns) {

    printf("Matrix:\n");

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

        printf("\n");

    }

}

// Function to calculate and display the sum of each row

void sumOfRows(int matrix[][100], int rows, int columns) {

    printf("\nSum of each row:\n");

    for (int i = 0; i < rows; i++) {

        int rowSum = 0;

        for (int j = 0; j < columns; j++) {

            rowSum += matrix[i][j];

        }

        printf("Row %d: %d\n", i + 1, rowSum);

    }

}

// Function to calculate and display the sum of each column

void sumOfColumns(int matrix[][100], int rows, int columns) {

    printf("\nSum of each column:\n");

    for (int j = 0; j < columns; j++) {

        int colSum = 0;

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

            colSum += matrix[i][j];

        }

        printf("Column %d: %d\n", j + 1, colSum);

    }

}

int main() {

    int rows, columns;

    // Input the number of rows and columns for the matrix

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    printf("Enter the number of columns: ");

    scanf("%d", &columns);

    // Check if the matrix dimensions are valid

    if (rows <= 0 || columns <= 0) {

        printf("Please enter valid dimensions for the matrix.\n");

        return 1;

    }

    // Declare a matrix

    int matrix[100][100];

    // Input elements for the matrix

    printf("\nFor the matrix:\n");

    inputMatrix(matrix, rows, columns);

    // Display the original matrix

    printf("\nOriginal Matrix:\n");

    displayMatrix(matrix, rows, columns);

    // Calculate and display the sum of each row

    sumOfRows(matrix, rows, columns);

    // Calculate and display the sum of each column

    sumOfColumns(matrix, rows, columns);

    return 0;

}

///15. Write a program to perform multiplication of two matrices.

#include <stdio.h>

// Function to input elements of a matrix

void inputMatrix(int matrix[][100], int rows, int columns) {

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

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

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

        }

    }

}

// Function to display a matrix

void displayMatrix(int matrix[][100], int rows, int columns) {

    printf("Matrix:\n");

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

        printf("\n");

    }

}

// Function to multiply two matrices

void multiplyMatrices(int matrix1[][100], int rows1, int columns1, int matrix2[][100], int rows2, int columns2, int result[][100]) {

    // Check if matrices can be multiplied

    if (columns1 != rows2) {

        printf("Matrices cannot be multiplied. Invalid dimensions.\n");

        return;

    }

    // Initialize the result matrix with zeros

    for (int i = 0; i < rows1; i++) {

        for (int j = 0; j < columns2; j++) {

            result[i][j] = 0;

        }

    }

    // Perform matrix multiplication

    for (int i = 0; i < rows1; i++) {

        for (int j = 0; j < columns2; j++) {

            for (int k = 0; k < columns1; k++) {

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

            }

        }

    }

}

int main() {

    int rows1, columns1, rows2, columns2;

    // Input the dimensions of the first matrix

    printf("Enter the number of rows for the first matrix: ");

    scanf("%d", &rows1);

    printf("Enter the number of columns for the first matrix: ");

    scanf("%d", &columns1);

    // Input the dimensions of the second matrix

    printf("Enter the number of rows for the second matrix: ");

    scanf("%d", &rows2);

    printf("Enter the number of columns for the second matrix: ");

    scanf("%d", &columns2);

    // Check if matrix dimensions are valid for multiplication

    if (columns1 != rows2) {

        printf("Matrices cannot be multiplied. Invalid dimensions.\n");

        return 1;

    }

    // Declare two matrices and the result matrix

    int matrix1[100][100], matrix2[100][100], result[100][100];

    // Input elements for the first matrix

    printf("\nFor the first matrix:\n");

    inputMatrix(matrix1, rows1, columns1);

    // Input elements for the second matrix

    printf("\nFor the second matrix:\n");

    inputMatrix(matrix2, rows2, columns2);

    // Perform matrix multiplication

    multiplyMatrices(matrix1, rows1, columns1, matrix2, rows2, columns2, result);

    // Display the result of multiplication

    printf("\nResult of Matrix Multiplication:\n");

    displayMatrix(result, rows1, columns2);

    return 0;

}

///16. Write a program to check whether a matrix is identity matrix or not.

#include <stdio.h>

// Function to input elements of a matrix

void inputMatrix(int matrix[][100], int rows, int columns) {

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

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

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

        }

    }

}

// Function to display a matrix

void displayMatrix(int matrix[][100], int rows, int columns) {

    printf("Matrix:\n");

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

        printf("\n");

    }

}

// Function to check if a matrix is an identity matrix

int isIdentityMatrix(int matrix[][100], int size) {

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++) {

            // Check diagonal elements for 1 and non-diagonal elements for 0

            if ((i == j && matrix[i][j] != 1) || (i != j && matrix[i][j] != 0)) {

                return 0; // Not an identity matrix

            }

        }

    }

    return 1; // Identity matrix

}

int main() {

    int size;

    // Input the size of the square matrix

    printf("Enter the size of the square matrix: ");

    scanf("%d", &size);

    // Check if the size is valid

    if (size <= 0) {

        printf("Please enter a valid size for the matrix.\n");

        return 1;

    }

    // Declare a matrix

    int matrix[100][100];

    // Input elements for the matrix

    printf("\nFor the matrix:\n");

    inputMatrix(matrix, size, size);

    // Display the original matrix

    printf("\nOriginal Matrix:\n");

    displayMatrix(matrix, size, size);

    // Check if the matrix is an identity matrix

    if (isIdentityMatrix(matrix, size)) {

        printf("\nThe matrix is an Identity Matrix.\n");

    } else {

        printf("\nThe matrix is not an Identity Matrix.\n");

    }

    return 0;

}

///17. Write a program to check whether a matrix is sparse matrix or not.

#include <stdio.h>

// Function to input elements of a matrix

void inputMatrix(int matrix[][100], int rows, int columns) {

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

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

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

        }

    }

}

// Function to display a matrix

void displayMatrix(int matrix[][100], int rows, int columns) {

    printf("Matrix:\n");

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

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

        }

        printf("\n");

    }

}

// Function to check if a matrix is a sparse matrix

int isSparseMatrix(int matrix[][100], int rows, int columns) {

    int zeroCount = 0;

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < columns; j++) {

            if (matrix[i][j] == 0) {

                zeroCount++;

            }

        }

    }

    // If the number of zero elements is greater than half the total elements, it's considered sparse

    return (zeroCount > (rows \* columns) / 2);

}

int main() {

    int rows, columns;

    // Input the dimensions of the matrix

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    printf("Enter the number of columns: ");

    scanf("%d", &columns);

    // Check if the matrix dimensions are valid

    if (rows <= 0 || columns <= 0) {

        printf("Please enter valid dimensions for the matrix.\n");

        return 1;

    }

    // Declare a matrix

    int matrix[100][100];

    // Input elements for the matrix

    printf("\nFor the matrix:\n");

    inputMatrix(matrix, rows, columns);

    // Display the original matrix

    printf("\nOriginal Matrix:\n");

    displayMatrix(matrix, rows, columns);

    // Check if the matrix is a sparse matrix

    if (isSparseMatrix(matrix, rows, columns)) {

        printf("\nThe matrix is a Sparse Matrix.\n");

    } else {

        printf("\nThe matrix is not a Sparse Matrix.\n");

    }

    return 0;

}

ASSIGNMENT 8

FILE

///1. Write a C Program to list all files and sub-directories in a directory.

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

void listFilesAndDirectories(const char \*path) {

    DIR \*dir;

    struct dirent \*entry;

    // Open the directory

    dir = opendir(path);

    // Check if the directory is opened successfully

    if (dir == NULL) {

        perror("Error opening directory");

        exit(EXIT\_FAILURE);

    }

    // Read directory entries

    while ((entry = readdir(dir)) != NULL) {

        printf("%s\n", entry->d\_name);

    }

    // Close the directory

    closedir(dir);

}

int main() {

    char path[100];

    // Get the directory path from the user

    printf("Enter the directory path: ");

    scanf("%s", path);

    // List files and sub-directories

    printf("\nFiles and sub-directories in '%s':\n", path);

    listFilesAndDirectories(path);

    return 0;

}

///2. Write a C Program to count number of lines in a file.

#include <stdio.h>

#include <stdlib.h>

int countLines(const char \*filename) {

    FILE \*file;

    char ch;

    int lines = 0;

    // Open the file

    file = fopen(filename, "r");

    // Check if the file is opened successfully

    if (file == NULL) {

        perror("Error opening file");

        exit(EXIT\_FAILURE);

    }

    // Count lines in the file

    while ((ch = fgetc(file)) != EOF) {

        if (ch == '\n') {

            lines++;

        }

    }

    // Close the file

    fclose(file);

    return lines;

}

int main() {

    char filename[100];

    // Get the filename from the user

    printf("Enter the filename: ");

    scanf("%s", filename);

    // Count the number of lines in the file

    int lines = countLines(filename);

    // Display the result

    printf("Number of lines in %s: %d\n", filename, lines);

    return 0;

}

///3. Write a C Program to print contents of file.

#include <stdio.h>

#include <stdlib.h>

void printFileContents(const char \*filename) {

    FILE \*file;

    char ch;

    // Open the file

    file = fopen(filename, "r");

    // Check if the file is opened successfully

    if (file == NULL) {

        perror("Error opening file");

        exit(EXIT\_FAILURE);

    }

    // Print contents of the file

    while ((ch = fgetc(file)) != EOF) {

        putchar(ch);

    }

    // Close the file

    fclose(file);

}

int main() {

    char filename[100];

    // Get the filename from the user

    printf("Enter the filename: ");

    scanf("%s", filename);

    // Print the contents of the file

    printf("\nContents of %s:\n", filename);

    printFileContents(filename);

    return 0;

}

///4. Write a C Program to copy contents of one file to another file.

#include <stdio.h>

#include <stdlib.h>

void copyFileContents(const char \*sourceFilename, const char \*destinationFilename) {

    FILE \*sourceFile, \*destinationFile;

    char ch;

    // Open the source file for reading

    sourceFile = fopen(sourceFilename, "r");

    // Check if the source file is opened successfully

    if (sourceFile == NULL) {

        perror("Error opening source file");

        exit(EXIT\_FAILURE);

    }

    // Open the destination file for writing

    destinationFile = fopen(destinationFilename, "w");

    // Check if the destination file is opened successfully

    if (destinationFile == NULL) {

        perror("Error opening destination file");

        fclose(sourceFile);

        exit(EXIT\_FAILURE);

    }

    // Copy contents from source file to destination file

    while ((ch = fgetc(sourceFile)) != EOF) {

        fputc(ch, destinationFile);

    }

    // Close both files

    fclose(sourceFile);

    fclose(destinationFile);

}

int main() {

    char sourceFilename[100], destinationFilename[100];

    // Get the source filename from the user

    printf("Enter the source filename: ");

    scanf("%s", sourceFilename);

    // Get the destination filename from the user

    printf("Enter the destination filename: ");

    scanf("%s", destinationFilename);

    // Copy contents from source file to destination file

    copyFileContents(sourceFilename, destinationFilename);

    printf("Contents of %s copied to %s successfully.\n", sourceFilename, destinationFilename);

    return 0;

}

///5. Write a C Program to merge contents of two files into a third file.

#include <stdio.h>

#include <stdlib.h>

void mergeFiles(const char \*file1, const char \*file2, const char \*mergedFile) {

    FILE \*sourceFile1, \*sourceFile2, \*mergedFilePtr;

    char ch;

    // Open the first source file for reading

    sourceFile1 = fopen(file1, "r");

    // Check if the first source file is opened successfully

    if (sourceFile1 == NULL) {

        perror("Error opening first source file");

        exit(EXIT\_FAILURE);

    }

    // Open the second source file for reading

    sourceFile2 = fopen(file2, "r");

    // Check if the second source file is opened successfully

    if (sourceFile2 == NULL) {

        perror("Error opening second source file");

        fclose(sourceFile1);

        exit(EXIT\_FAILURE);

    }

    // Open the merged file for writing

    mergedFilePtr = fopen(mergedFile, "w");

    // Check if the merged file is opened successfully

    if (mergedFilePtr == NULL) {

        perror("Error opening merged file");

        fclose(sourceFile1);

        fclose(sourceFile2);

        exit(EXIT\_FAILURE);

    }

    // Copy contents from the first source file to the merged file

    while ((ch = fgetc(sourceFile1)) != EOF) {

        fputc(ch, mergedFilePtr);

    }

    // Copy contents from the second source file to the merged file

    while ((ch = fgetc(sourceFile2)) != EOF) {

        fputc(ch, mergedFilePtr);

    }

    // Close all files

    fclose(sourceFile1);

    fclose(sourceFile2);

    fclose(mergedFilePtr);

}

int main() {

    char sourceFile1[100], sourceFile2[100], mergedFile[100];

    // Get the first source filename from the user

    printf("Enter the first source filename: ");

    scanf("%s", sourceFile1);

    // Get the second source filename from the user

    printf("Enter the second source filename: ");

    scanf("%s", sourceFile2);

    // Get the merged filename from the user

    printf("Enter the merged filename: ");

    scanf("%s", mergedFile);

    // Merge contents from two source files to the merged file

    mergeFiles(sourceFile1, sourceFile2, mergedFile);

    printf("Contents of %s and %s merged into %s successfully.\n", sourceFile1, sourceFile2, mergedFile);

    return 0;

}

///6. Write a C program to delete a file.

#include <stdio.h>

#include <stdlib.h>

int main() {

    char filename[100];

    // Get the filename from the user

    printf("Enter the filename to delete: ");

    scanf("%s", filename);

    // Attempt to delete the file

    if (remove(filename) == 0) {

        printf("File '%s' deleted successfully.\n", filename);

    } else {

        perror("Error deleting file");

        exit(EXIT\_FAILURE);

    }

    return 0;

}

FUNCTION

///1. Write a C program to add, subtract, multiply and divide two integers using userdefined type function with return type.

#include <stdio.h>

// Function to add two integers

int add(int num1, int num2) {

    return num1 + num2;

}

// Function to subtract two integers

int subtract(int num1, int num2) {

    return num1 - num2;

}

// Function to multiply two integers

int multiply(int num1, int num2) {

    return num1 \* num2;

}

// Function to divide two integers

float divide(int num1, int num2) {

    // Check if the divisor is not zero to avoid division by zero

    if (num2 != 0) {

        return (float)num1 / num2; // Convert the result to float for accurate division

    } else {

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

        return 0.0; // Return 0 in case of division by zero

    }

}

int main() {

    int num1, num2;

    // Input two integers from the user

    printf("Enter first integer: ");

    scanf("%d", &num1);

 printf("Enter second integer: ");

    scanf("%d", &num2);

    // Perform operations and display the results

    printf("Sum: %d\n", add(num1, num2));

    printf("Difference: %d\n", subtract(num1, num2));

    printf("Product: %d\n", multiply(num1, num2));

    // For division, display the result only if it's not division by zero

    float result = divide(num1, num2);

    if (result != 0.0) {

        printf("Quotient: %.2f\n", result);

    }

    return 0;

}

///2. Write a C program to calculate sum of first 20 natural numbers using recursive function.

#include <stdio.h>

// Recursive function to calculate the sum of first n natural numbers

int sumOfNaturalNumbers(int n) {

    // Base case: if n is 0, the sum is 0

    if (n == 0) {

        return 0;

    } else {

        // Recursive case: sum of n natural numbers is n + sum of (n-1) natural numbers

        return n + sumOfNaturalNumbers(n - 1);

    }

}

int main() {

    // Calculate the sum of the first 20 natural numbers

    int n = 20;

    int result = sumOfNaturalNumbers(n);

    // Display the result

    printf("Sum of first %d natural numbers: %d\n", n, result);

    return 0;

}

///3. Write a C program to generate Fibonacci series using recursive function.

#include <stdio.h>

// Recursive function to generate Fibonacci series

int fibonacci(int n) {

    if (n <= 1) {

        return n;

    } else {

        // Fibonacci series: F(n) = F(n-1) + F(n-2)

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

    }

}

int main() {

    int n;

    // Input the number of terms in the Fibonacci series

    printf("Enter the number of terms in the Fibonacci series: ");

    scanf("%d", &n);

    // Display the Fibonacci series

    printf("Fibonacci Series: ");

    for (int i = 0; i < n; ++i) {

        printf("%d ", fibonacci(i));

    }

    return 0;

}

///4. Write a C program to swap two integers using call by value and call by reference methods of passing arguments to a function.

#include <stdio.h>

// Function to swap two integers using call by value

void swapByValue(int a, int b) {

    int temp = a;

    a = b;

    b = temp;

}

// Function to swap two integers using call by reference

void swapByReference(int \*a, int \*b) {

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

int main() {

    int num1, num2;

    // Input two integers from the user

    printf("Enter the first integer: ");

    scanf("%d", &num1);

    printf("Enter the second integer: ");

    scanf("%d", &num2);

    // Display the original values

    printf("Original values: num1 = %d, num2 = %d\n", num1, num2);

    // Swap using call by value (no change in main)

    swapByValue(num1, num2);

    printf("After swap by value: num1 = %d, num2 = %d\n", num1, num2);

    // Swap using call by reference (changes reflected in main)

    swapByReference(&num1, &num2);

    printf("After swap by reference: num1 = %d, num2 = %d\n", num1, num2);

    return 0;

}

///5. Write a C program to find sum of digits of the number using Recursive Function.

#include <stdio.h>

// Recursive function to find the sum of digits

int sumOfDigits(int n) {

    // Base case: if the number is 0, the sum is 0

    if (n == 0) {

        return 0;

    } else {

        // Recursive case: sum of digits is the last digit + sum of digits in the remaining number

        return n % 10 + sumOfDigits(n / 10);

    }

}

int main() {

    int number;

    // Input a number from the user

    printf("Enter a number: ");

    scanf("%d", &number);

    // Calculate the sum of digits using the recursive function

    int result = sumOfDigits(number);

    // Display the result

    printf("Sum of digits of %d: %d\n", number, result);

    return 0;

}

///6. Write a C program to read an integer number and print the reverse of that number using recursion.

#include <stdio.h>

// Function to reverse a given number

int reverseNumber(int n) {

    int reversedNumber = 0;

    while (n != 0) {

        reversedNumber = reversedNumber \* 10 + n % 10;

        n /= 10;

    }

    return reversedNumber;

}

int main() {

    int number;

    // Input an integer from the user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Call the function to reverse the number

    int reversed = reverseNumber(number);

    // Display the reversed number

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

    return 0;

}

///7. Write a C program to find maximum and minimum between two numbers using functions.

#include <stdio.h>

// Function to find the maximum between two numbers

int findMaximum(int num1, int num2) {

    return (num1 > num2) ? num1 : num2;

}

// Function to find the minimum between two numbers

int findMinimum(int num1, int num2) {

    return (num1 < num2) ? num1 : num2;

}

int main() {

    int num1, num2;

    // Input two numbers from the user

    printf("Enter the first number: ");

    scanf("%d", &num1);

    printf("Enter the second number: ");

    scanf("%d", &num2);

    // Call the functions to find maximum and minimum

    int maximum = findMaximum(num1, num2);

    int minimum = findMinimum(num1, num2);

    // Display the results

    printf("Maximum: %d\n", maximum);

    printf("Minimum: %d\n", minimum);

    return 0;

}

///8. Write a C program to check whether a number is even or odd using functions.

#include <stdio.h>

// Function to check whether a number is even or odd

void checkEvenOdd(int num) {

    if (num % 2 == 0) {

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

    } else {

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

    }

}

int main() {

    int number;

    // Input a number from the user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Call the function to check even or odd

    checkEvenOdd(number);

    return 0;

}

///9. Write a C program to check whether a number is prime, Armstrong or perfect number using functions.

#include <stdio.h>

#include <math.h>

// Function to check if a number is prime

int isPrime(int num) {

    if (num < 2) {

        return 0; // Not a prime number

    }

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

        if (num % i == 0) {

            return 0; // Not a prime number

        }

    }

    return 1; // Prime number

}

// Function to check if a number is Armstrong

int isArmstrong(int num) {

    int originalNum = num;

    int sum = 0, digit;

    while (num > 0) {

        digit = num % 10;

        sum += pow(digit, 3); // For Armstrong number of 3 digits

        num /= 10;

    }

    return (sum == originalNum);

}

// Function to check if a number is perfect

int isPerfect(int num) {

    int sum = 1; // Start with 1 as every number is divisible by 1

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

        if (num % i == 0) {

            sum += i;

        }

    }

    return (sum == num);

}

int main() {

    int number;

    // Input a number from the user

    printf("Enter an integer: ");

    scanf("%d", &number);

    // Check if the number is prime

    if (isPrime(number)) {

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

    } else {

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

    }

    // Check if the number is Armstrong

    if (isArmstrong(number)) {

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

    } else {

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

    }

    // Check if the number is perfect

    if (isPerfect(number)) {

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

    } else {

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

    }

    return 0;

}

///10. Write a C program to find power of any number using recursion.

#include <stdio.h>

// Recursive function to calculate power

double power(double base, int exponent) {

    // Base case: if the exponent is 0, the result is 1

    if (exponent == 0) {

        return 1.0;

    } else if (exponent > 0) {

        // Recursive case: power is base \* power(base, exponent - 1)

        return base \* power(base, exponent - 1);

    } else {

        // Handling negative exponent by taking reciprocal

        return 1.0 / (base \* power(base, -exponent - 1));

    }

}

int main() {

    double base;

    int exponent;

    // Input base and exponent from the user

    printf("Enter the base: ");

    scanf("%lf", &base);

    printf("Enter the exponent: ");

    scanf("%d", &exponent);

    // Call the recursive function to calculate power

    double result = power(base, exponent);

    // Display the result

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

    return 0;

}

POINTER

///1. Write a C program to find the sum of all the elements of an array using pointers.

#include <stdio.h>

// Function to find the sum of array elements using pointers

int findSum(int \*arr, int size) {

    int sum = 0;

    // Iterate through the array using pointers

    for (int i = 0; i < size; ++i) {

        sum += \*(arr + i); // Equivalent to arr[i]

    }

    return sum;

}

int main() {

    int size;

    // Get the size of the array from the user

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    int arr[size];

    // Get array elements from the user

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

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Calculate the sum using pointers

    int sum = findSum(arr, size);

    // Display the sum

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

    return 0;

}

///2. Write a C program to swap value of two variables using pointer.

#include <stdio.h>

// Function to swap values using pointers

void swap(int \*a, int \*b) {

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

int main() {

    int num1, num2;

    // Get values from the user

    printf("Enter value for num1: ");

    scanf("%d", &num1);

    printf("Enter value for num2: ");

    scanf("%d", &num2);

    // Display the values before swapping

    printf("\nValues before swapping:\n");

    printf("num1: %d\n", num1);

    printf("num2: %d\n", num2);

    // Swap values using the swap function

    swap(&num1, &num2);

    // Display the values after swapping

    printf("\nValues after swapping:\n");

    printf("num1: %d\n", num1);

    printf("num2: %d\n", num2);

    return 0;

}

///3. Write a C program to add two numbers using pointers.

#include <stdio.h>

// Function to add two numbers using pointers

int addNumbers(int \*a, int \*b) {

    return (\*a + \*b);

}

int main() {

    int num1, num2;

    // Get values from the user

    printf("Enter the first number: ");

    scanf("%d", &num1);

    printf("Enter the second number: ");

    scanf("%d", &num2);

    // Display the values before addition

    printf("\nValues before addition:\n");

    printf("num1: %d\n", num1);

    printf("num2: %d\n", num2);

    // Add the numbers using the addNumbers function

    int sum = addNumbers(&num1, &num2);

    // Display the sum

    printf("\nSum of the two numbers: %d\n", sum);

    return 0;

}

///4. Write a C program to input and print array elements using pointer.

#include <stdio.h>

// Function to input array elements using pointers

void inputArray(int \*arr, int size) {

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

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", (arr + i)); // Equivalent to &arr[i]

    }

}

// Function to print array elements using pointers

void printArray(int \*arr, int size) {

    printf("\nArray elements:\n");

    for (int i = 0; i < size; ++i) {

        printf("%d ", \*(arr + i)); // Equivalent to arr[i]

    }

    printf("\n");

}

int main() {

    int size;

    // Get the size of the array from the user

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    int arr[size];

    // Input array elements using pointers

    inputArray(arr, size);

    // Print array elements using pointers

    printArray(arr, size);

    return 0;

}

///5. Write a C program to copy one array to another using pointer.

#include <stdio.h>

// Function to copy one array to another using pointers

void copyArray(int \*source, int \*destination, int size) {

    for (int i = 0; i < size; ++i) {

        \*(destination + i) = \*(source + i); // Equivalent to destination[i] = source[i]

    }

}

// Function to print array elements using pointers

void printArray(int \*arr, int size) {

    printf("\nArray elements:\n");

    for (int i = 0; i < size; ++i) {

        printf("%d ", \*(arr + i)); // Equivalent to arr[i]

    }

    printf("\n");

}

int main() {

    int size;

    // Get the size of the array from the user

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    int sourceArray[size];

    int destinationArray[size];

    // Input elements into the source array

    printf("Enter the elements of the source array:\n");

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", &sourceArray[i]);

    }

    // Copy elements from source array to destination array using pointers

    copyArray(sourceArray, destinationArray, size);

    // Print the source and destination arrays

    printf("\nSource Array:\n");

    printArray(sourceArray, size);

    printf("\nDestination Array after copying:\n");

    printArray(destinationArray, size);

    return 0;

}

///6. Write a C program to swap two arrays using pointers.

#include <stdio.h>

// Function to swap two arrays using pointers

void swapArrays(int \*arr1, int \*arr2, int size) {

    for (int i = 0; i < size; ++i) {

        // Swap elements at each position between arr1 and arr2

        int temp = \*(arr1 + i);

        \*(arr1 + i) = \*(arr2 + i);

        \*(arr2 + i) = temp;

    }

}

// Function to print array elements using pointers

void printArray(int \*arr, int size) {

    printf("\nArray elements:\n");

    for (int i = 0; i < size; ++i) {

        printf("%d ", \*(arr + i));

    }

    printf("\n");

}

int main() {

    int size;

    // Get the size of the arrays from the user

    printf("Enter the size of the arrays: ");

    scanf("%d", &size);

    int arr1[size];

    int arr2[size];

    // Input elements into the first array

    printf("Enter the elements of the first array:\n");

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr1[i]);

    }

    // Input elements into the second array

    printf("Enter the elements of the second array:\n");

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr2[i]);

    }

    // Print the arrays before swapping

    printf("\nArrays before swapping:\n");

    printf("Array 1:\n");

    printArray(arr1, size);

    printf("Array 2:\n");

    printArray(arr2, size);

    // Swap the arrays using pointers

    swapArrays(arr1, arr2, size);

    // Print the arrays after swapping

    printf("\nArrays after swapping:\n");

    printf("Array 1:\n");

    printArray(arr1, size);

    printf("Array 2:\n");

    printArray(arr2, size);

    return 0;

}

///7. Write a C program to reverse an array using pointers.

#include <stdio.h>

// Function to reverse an array using pointers

void reverseArray(int \*arr, int size) {

    int \*start = arr;

    int \*end = arr + size - 1;

    while (start < end) {

        // Swap elements at the start and end positions

        int temp = \*start;

        \*start = \*end;

        \*end = temp;

        // Move pointers towards each other

        start++;

        end--;

    }

}

// Function to print array elements using pointers

void printArray(int \*arr, int size) {

    printf("\nArray elements:\n");

    for (int i = 0; i < size; ++i) {

        printf("%d ", \*(arr + i));

    }

    printf("\n");

}

int main() {

    int size;

    // Get the size of the array from the user

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    int arr[size];

    // Input elements into the array

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

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Print the array before reversing

    printf("\nArray before reversing:\n");

    printArray(arr, size);

    // Reverse the array using pointers

    reverseArray(arr, size);

    // Print the array after reversing

    printf("\nArray after reversing:\n");

    printArray(arr, size);

    return 0;

}

///8. Write a C program to search an element in array using pointers.

#include <stdio.h>

// Function to search for an element in an array using pointers

int searchElement(int \*arr, int size, int key) {

    for (int i = 0; i < size; ++i) {

        if (\*(arr + i) == key) {

            // Element found, return its position (1-based index)

            return i + 1;

        }

    }

    // Element not found, return -1

    return -1;

}

// Function to print array elements using pointers

void printArray(int \*arr, int size) {

    printf("\nArray elements:\n");

    for (int i = 0; i < size; ++i) {

        printf("%d ", \*(arr + i));

    }

    printf("\n");

}

int main() {

    int size, key;

    // Get the size of the array from the user

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    int arr[size];

    // Input elements into the array

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

    for (int i = 0; i < size; ++i) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    // Print the array

    printArray(arr, size);

    // Get the element to search from the user

    printf("\nEnter the element to search: ");

    scanf("%d", &key);

    // Search for the element using pointers

    int position = searchElement(arr, size, key);

    // Display the result

    if (position != -1) {

        printf("\nElement found at position %d.\n", position);

    } else {

        printf("\nElement not found in the array.\n");

    }

    return 0;

}

///9. Write a C program to add two 2 X 2 matrix using pointers.

#include <stdio.h>

// Function to add two matrices using pointers

void addMatrices(int mat1[2][2], int mat2[2][2], int result[2][2]) {

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

            \*(\*(result + i) + j) = \*(\*(mat1 + i) + j) + \*(\*(mat2 + i) + j);

        }

    }

}

// Function to print a 2x2 matrix using pointers

void printMatrix(int mat[2][2]) {

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

            printf("%d ", \*(\*(mat + i) + j));

        }

        printf("\n");

    }

}

int main() {

    int mat1[2][2], mat2[2][2], result[2][2];

    // Input elements for the first matrix

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

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

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

            scanf("%d", &mat1[i][j]);

        }

    }

    // Input elements for the second matrix

    printf("\nEnter elements for the second matrix:\n");

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

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

            scanf("%d", &mat2[i][j]);

        }

    }

    // Add the matrices using pointers

    addMatrices(mat1, mat2, result);

    // Display the matrices and the result

    printf("\nMatrix 1:\n");

    printMatrix(mat1);

    printf("\nMatrix 2:\n");

    printMatrix(mat2);

    printf("\nSum of the matrices:\n");

    printMatrix(result);

    return 0;

}

///10. Write a C program to multiply two 2 X 2 matrix using pointers.

#include <stdio.h>

// Function to multiply two matrices using pointers

void multiplyMatrices(int mat1[2][2], int mat2[2][2], int result[2][2]) {

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

            \*(\*(result + i) + j) = 0;  // Initialize result matrix element to 0

            // Multiply and accumulate the product for each element

            for (int k = 0; k < 2; ++k) {

                \*(\*(result + i) + j) += \*(\*(mat1 + i) + k) \* \*(\*(mat2 + k) + j);

            }

        }

    }

}

// Function to print a 2x2 matrix using pointers

void printMatrix(int mat[2][2]) {

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

            printf("%d ", \*(\*(mat + i) + j));

        }

        printf("\n");

    }

}

int main() {

    int mat1[2][2], mat2[2][2], result[2][2];

    // Input elements for the first matrix

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

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

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

            scanf("%d", &mat1[i][j]);

        }

    }

    // Input elements for the second matrix

    printf("\nEnter elements for the second matrix:\n");

    for (int i = 0; i < 2; ++i) {

        for (int j = 0; j < 2; ++j) {

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

            scanf("%d", &mat2[i][j]);

        }

    }

    // Multiply the matrices using pointers

    multiplyMatrices(mat1, mat2, result);

    // Display the matrices and the result

    printf("\nMatrix 1:\n");

    printMatrix(mat1);

    printf("\nMatrix 2:\n");

    printMatrix(mat2);

    printf("\nProduct of the matrices:\n");

    printMatrix(result);

    return 0;

}

///11. Write a C program to find length of string using pointers.

#include <stdio.h>

// Function to find the length of a string using pointers

int stringLength(char \*str) {

    int length = 0;

    // Iterate through the string using pointers until the null character is encountered

    while (\*str != '\0') {

        length++;

        str++;

    }

    return length;

}

int main() {

    char inputString[100];

    // Get a string from the user

    printf("Enter a string: ");

    fgets(inputString, sizeof(inputString), stdin);

    // Calculate the length of the string using pointers

    int length = stringLength(inputString);

    // Display the length of the string

    printf("Length of the string: %d\n", length);

    return 0;

}

///12. Write a C program to copy one string to another using pointer.

#include <stdio.h>

// Function to copy one string to another using pointers

void copyString(char \*source, char \*destination) {

    // Iterate through the characters using pointers until the null character is encountered

    while (\*source != '\0') {

        \*destination = \*source;

        source++;

        destination++;

    }

    // Add null character to the end of the destination string

    \*destination = '\0';

}

int main() {

    char sourceString[100];

    char destinationString[100];

    // Get a string from the user

    printf("Enter a string: ");

    fgets(sourceString, sizeof(sourceString), stdin);

    // Copy the string using pointers

    copyString(sourceString, destinationString);

    // Display the copied string

    printf("Copied string: %s\n", destinationString);

    return 0;

}

///13. Write a C program to concatenate two strings using pointers.

#include <stdio.h>

// Function to concatenate two strings using pointers

void concatenateStrings(char \*str1, char \*str2, char \*result) {

    // Copy the characters of the first string to the result string

    while (\*str1 != '\0') {

        \*result = \*str1;

        str1++;

        result++;

    }

    // Copy the characters of the second string to the result string

    while (\*str2 != '\0') {

        \*result = \*str2;

        str2++;

        result++;

    }

    // Add null character to the end of the result string

    \*result = '\0';

}

int main() {

    char firstString[100];

    char secondString[100];

    char resultString[200];

    // Get the first string from the user

    printf("Enter the first string: ");

    fgets(firstString, sizeof(firstString), stdin);

    // Get the second string from the user

    printf("Enter the second string: ");

    fgets(secondString, sizeof(secondString), stdin);

    // Concatenate the strings using pointers

    concatenateStrings(firstString, secondString, resultString);

    // Display the concatenated string

    printf("Concatenated string: %s\n", resultString);

    return 0;

}

///14. Write a C program to compare two strings using pointers.

#include <stdio.h>

// Function to compare two strings using pointers

int compareStrings(char \*str1, char \*str2) {

    // Compare each character of the strings using pointers

    while (\*str1 != '\0' && \*str2 != '\0') {

        if (\*str1 != \*str2) {

            // If characters are not equal, return the ASCII difference

            return \*str1 - \*str2;

        }

        str1++;

        str2++;

    }

    // Strings are equal if both end at the null character

    return \*str1 - \*str2;

}

int main() {

    char firstString[100];

    char secondString[100];

    // Get the first string from the user

    printf("Enter the first string: ");

    fgets(firstString, sizeof(firstString), stdin);

    // Get the second string from the user

    printf("Enter the second string: ");

    fgets(secondString, sizeof(secondString), stdin);

    // Compare the strings using pointers

    int result = compareStrings(firstString, secondString);

    // Display the result of the comparison

    if (result == 0) {

        printf("The strings are equal.\n");

    } else if (result < 0) {

        printf("The first string is less than the second string.\n");

    } else {

        printf("The first string is greater than the second string.\n");

    }

    return 0;

}

STRING

///1. Write a C program to compare two strings.

#include <stdio.h>

#include <string.h>

int main() {

    char str1[100], str2[100];

    // Input from user

    printf("Enter the first string: ");

    scanf("%s", str1);

    printf("Enter the second string: ");

    scanf("%s", str2);

    // Compare the strings using strcmp function

    int result = strcmp(str1, str2);

    // Check the result and print the comparison result

    if (result == 0) {

        printf("The strings are equal.\n");

    } else if (result < 0) {

        printf("The first string is lexicographically smaller than the second string.\n");

    } else {

        printf("The first string is lexicographically larger than the second string.\n");

    }

    return 0;

}

///2. Write a C program to concatenate two strings.

#include <stdio.h>

#include <string.h>

int main() {

    char str1[100], str2[100];

    // Input from user

    printf("Enter the first string: ");

    scanf("%s", str1);

    printf("Enter the second string: ");

    scanf("%s", str2);

    // Concatenate the strings using strcat function

    strcat(str1, str2);

    // Print the concatenated string

    printf("Concatenated string: %s\n", str1);

    return 0;

}

///3. Write a C program to print the reverse of a string.

#include <stdio.h>

#include <string.h>

int main() {

    char str[100];

    // Input from user

    printf("Enter a string: ");

    scanf("%s", str);

    // Calculate the length of the string

    int length = strlen(str);

    // Print the reverse of the string

    printf("Reverse of the string: ");

    for (int i = length - 1; i >= 0; i--) {

        printf("%c", str[i]);

    }

    printf("\n");

    return 0;

}

///4. Write a C program to find the 3rd element of a string.

#include <stdio.h>

#include <string.h>

int main() {

    char str[100];

    // Input from user

    printf("Enter a string: ");

    scanf("%s", str);

    // Check if the string has at least 3 characters

    if (strlen(str) >= 3) {

        // Print the third element (character) of the string

        printf("The third element of the string: %c\n", str[2]);

    } else {

        printf("The string has less than 3 characters.\n");

    }

    return 0;

}

STRUCTURE

///1. Write a C program to create a structure named company which has name, address, phone and noOfEmployee as member variables. Read name of company, its address, phone and noOfEmployee. Finally display these members‟ value.

#include <stdio.h>

// Define the structure named "company"

struct Company {

    char name[50];

    char address[100];

    char phone[15];

    int noOfEmployee;

};

int main() {

    // Declare a variable of type "struct Company"

    struct Company myCompany;

    // Input information about the company from the user

    printf("Enter the name of the company: ");

    fgets(myCompany.name, sizeof(myCompany.name), stdin);

    printf("Enter the address of the company: ");

    fgets(myCompany.address, sizeof(myCompany.address), stdin);

    printf("Enter the phone number of the company: ");

    fgets(myCompany.phone, sizeof(myCompany.phone), stdin);

    printf("Enter the number of employees in the company: ");

    scanf("%d", &myCompany.noOfEmployee);

    // Display the information about the company

    printf("\nCompany Information:\n");

    printf("Name: %s", myCompany.name);

    printf("Address: %s", myCompany.address);

    printf("Phone: %s", myCompany.phone);

    printf("Number of Employees: %d\n", myCompany.noOfEmployee);

    return 0;

}

///2. Define a structure “complex” (typedef) to read two complex numbers and perform addition, subtraction of these two complex numbers and display the result.

#include <stdio.h>

// Define a structure "complex"

typedef struct {

    float real;

    float imag;

} Complex;

// Function to add two complex numbers

Complex addComplex(Complex num1, Complex num2) {

    Complex result;

    result.real = num1.real + num2.real;

    result.imag = num1.imag + num2.imag;

    return result;

}

// Function to subtract two complex numbers

Complex subtractComplex(Complex num1, Complex num2) {

    Complex result;

    result.real = num1.real - num2.real;

    result.imag = num1.imag - num2.imag;

    return result;

}

int main() {

    Complex complexNum1, complexNum2, sum, difference;

    // Input the first complex number

    printf("Enter the real part of the first complex number: ");

    scanf("%f", &complexNum1.real);

    printf("Enter the imaginary part of the first complex number: ");

    scanf("%f", &complexNum1.imag);

    // Input the second complex number

    printf("Enter the real part of the second complex number: ");

    scanf("%f", &complexNum2.real);

    printf("Enter the imaginary part of the second complex number: ");

    scanf("%f", &complexNum2.imag);

    // Perform addition and subtraction

    sum = addComplex(complexNum1, complexNum2);

    difference = subtractComplex(complexNum1, complexNum2);

    // Display the results

    printf("\nSum: %.2f + %.2fi\n", sum.real, sum.imag);

    printf("Difference: %.2f - %.2fi\n", difference.real, difference.imag);

    return 0;

}

///3. Write a C program to read RollNo, Name, Address, Age & average-marks of 12 students in the BCT class and display the details from function.

#include <stdio.h>

// Structure to hold student information

struct Student {

    int rollNo;

    char name[50];

    char address[100];

    int age;

    float averageMarks;

};

// Function to display student details

void displayDetails(struct Student student) {

    printf("\nRoll No: %d", student.rollNo);

    printf("\nName: %s", student.name);

    printf("\nAddress: %s", student.address);

    printf("\nAge: %d", student.age);

    printf("\nAverage Marks: %.2f\n", student.averageMarks);

}

int main() {

    // Array to store information for 12 students

    struct Student students[12];

    // Reading details for each student

    for (int i = 0; i < 12; ++i) {

        printf("\nEnter details for student %d:\n", i + 1);

        printf("Roll No: ");

        scanf("%d", &students[i].rollNo);

        printf("Name: ");

        scanf("%s", students[i].name);

        printf("Address: ");

        scanf("%s", students[i].address);

        printf("Age: ");

        scanf("%d", &students[i].age);

        printf("Average Marks: ");

        scanf("%f", &students[i].averageMarks);

    }

    // Displaying details for each student using the function

    printf("\nDetails of BCT class students:\n");

    for (int i = 0; i < 12; ++i) {

        displayDetails(students[i]);

    }

    return 0;

}

///4. Write a C program to add two distances in feet and inches using structure

#include <stdio.h>

// Structure to represent distances

struct Distance {

    int feet;

    float inches;

};

// Function to add two distances

struct Distance addDistances(struct Distance d1, struct Distance d2) {

    struct Distance result;

    // Add feet and inches separately

    result.feet = d1.feet + d2.feet;

    result.inches = d1.inches + d2.inches;

    // Adjust inches if greater than or equal to 12

    if (result.inches >= 12.0) {

        result.inches -= 12.0;

        result.feet += 1;

    }

    return result;

}

// Function to display a distance

void displayDistance(struct Distance d) {

    printf("%d feet %.2f inches\n", d.feet, d.inches);

}

int main() {

    struct Distance distance1, distance2, sum;

    // Get values for the first distance from the user

    printf("Enter the first distance:\n");

    printf("Feet: ");

    scanf("%d", &distance1.feet);

    printf("Inches: ");

    scanf("%f", &distance1.inches);

    // Get values for the second distance from the user

    printf("\nEnter the second distance:\n");

    printf("Feet: ");

    scanf("%d", &distance2.feet);

    printf("Inches: ");

    scanf("%f", &distance2.inches);

    // Add the distances using the addDistances function

    sum = addDistances(distance1, distance2);

    // Display the result

    printf("\nSum of distances: ");

    displayDistance(sum);

    return 0;

}

///5. Write a C program to read and print an Employee’s Details using Structure.

#include <stdio.h>

// Structure to represent employee details

struct Employee {

    char name[50];

    int employeeId;

    float salary;

};

// Function to read employee details

void readEmployeeDetails(struct Employee \*emp) {

    printf("Enter employee details:\n");

    // Get employee name

    printf("Name: ");

    scanf("%s", emp->name);

    // Get employee ID

    printf("Employee ID: ");

    scanf("%d", &emp->employeeId);

    // Get employee salary

    printf("Salary: ");

    scanf("%f", &emp->salary);

}

// Function to print employee details

void printEmployeeDetails(struct Employee emp) {

    printf("\nEmployee details:\n");

    printf("Name: %s\n", emp.name);

    printf("Employee ID: %d\n", emp.employeeId);

    printf("Salary: %.2f\n", emp.salary);

}

int main() {

    // Declare an Employee structure

    struct Employee employee;

    // Read employee details using the readEmployeeDetails function

    readEmployeeDetails(&employee);

    // Print employee details using the printEmployeeDetails function

    printEmployeeDetails(employee);

    return 0;

}