

# LIST OF PROGRAMS AND SOLUTIONS

## Assignment – 1

### Topic: Control Structures

1. Write a C program to find sum and average of three numbers.
2. Write a C program to find the sum of individual digits of a given positive integer.
3. Write a C program to generate the first n terms of the Fibonacci sequence.
4. Write a C program to generate prime numbers between m to n where m and n are user inputs.
5. Write a C program to check whether a given number is an Armstrong Number or Not.
6. Write a C program to evaluate the algebraic expression  $(ax+b)/(ax-b)$ .
7. Write a C program to check if the given number is the perfect number.
8. Write a C program to check if the given number is the strong number.
9. Write a C program to print your name without using any semicolon in the program.
10. Write a C program to convert temperature in Celsius to Fahrenheit and vice-versa.
11. Write a C program to check whether a number is Palindrome or not.
12. Write a C program to find the maximum between two numbers.
13. Write a C program to find the maximum between three numbers.
14. Write a C program to check whether a number is negative, positive or zero.
15. Write a C program to check whether a number is divisible by 5 and 11 or not within the range 100 to 500.
16. Write a C program to check whether a number is even or odd.
17. Write a C program to check whether a year is a leap year or not.
18. Write a C program to check whether a character is alphabet or not.
19. Write a C program to input any alphabet and check whether it is vowel or consonant.
20. Write a C program to input any character and check whether it is an alphabet, digit or special character.

```
1 //
2 //1. Write a C program to find sum and average of three numbers.
3 //
4
5 #include <stdio.h>
6
7 int main() {
8     float num1, num2, num3, sum, average;
9
10    // Taking input
11    printf("Enter three numbers: ");
12    scanf("%f %f %f", &num1, &num2, &num3);
13
14    // Calculating sum and average
15    sum = num1 + num2 + num3;
16    average = sum / 3;
17
18    // Displaying the result
19    printf("Sum = %.2f\n", sum);
20    printf("Average = %.2f\n", average);
21
22    return 0;
23 }
24 /*
25 Enter three numbers: 10 20 30
26 Sum = 60.00
27 Average = 20.00
28 */
29
```

```
1 //
2 //2. Write a C program to find the sum of individual digits of a given positive integer.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int num, digit, sum = 0;
8
9     // Input from user
10    printf("Enter a positive integer: ");
11    scanf("%d", &num);
12
13    // Loop to extract and sum digits
14    while (num != 0) {
15        digit = num % 10;    // Get last digit
16        sum += digit;        // Add digit to sum
17        num = num / 10;      // Remove last digit
18    }
19
20    // Output the result
21    printf("Sum of digits = %d\n", sum);
22
23    return 0;
24 }
25
26 /*
27 Enter a positive integer: 456
28 Sum of digits = 15
29 */
```

```
1 //
2 //3. Write a C program to generate the first n terms of the Fibonacci sequence.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int n, i;
8     int a = 0, b = 1, next;
9
10    printf("Enter the number of terms: ");
11    scanf("%d", &n);
12
13    printf("Fibonacci Sequence: ");
14    for (i = 0; i < n; i++) {
15        printf("%d ", a);
16        next = a + b;
17        a = b;
18        b = next;
19    }
20
21    printf("\n");
22    return 0;
23 }
24 /*
25 Enter the number of terms: 7
26 Fibonacci Sequence: 0 1 1 2 3 5 8
27 */
```

```

1 //
2 //4. Write a C program to generate prime numbers between 1 to n.
3 //
4
5 #include <stdio.h>
6
7 // Function to check if a number is prime
8 int isPrime(int num) {
9     if (num <= 1)
10         return 0; // 0 and 1 are not prime
11     for (int i = 2; i * i <= num; i++) {
12         if (num % i == 0)
13             return 0;
14     }
15     return 1;
16 }
17
18 int main() {
19     int m, n, start, end;
20
21     // Get input from user
22     printf("Enter the first number (m): ");
23     scanf("%d", &m);
24     printf("Enter the second number (n): ");
25     scanf("%d", &n);
26
27     // Determine the correct range (start should be smaller)
28     if (m < n) {
29         start = m;
30         end = n;
31     } else {
32         start = n;
33         end = m;
34     }
35
36     printf("Prime numbers between %d and %d are:\n", start, end);
37     for (int i = start; i <= end; i++) {
38         if (isPrime(i)) {
39             printf("%d ", i);
40         }
41     }
42
43     printf("\n");
44     return 0;
45 }
46
47 /*
48 Enter the first number (m): 50
49 Enter the second number (n): 30
50 Prime numbers between 30 and 50 are:
51 31 37 41 43 47
52 */

```

```

1 //
2 // 5. Write a C program to check whether a given number is an Armstrong Number or Not.
3 //
4 #include <stdio.h>
5 #include <math.h>
6
7 int main() {
8     int num, originalNum, remainder, result = 0, n = 0;
9
10    // Get input from user
11    printf("Enter a number: ");
12    scanf("%d", &num);
13
14    originalNum = num;
15
16    // Count the number of digits
17    while (originalNum != 0) {
18        originalNum /= 10;
19        n++;
20    }
21
22    originalNum = num;
23
24    // Calculate the sum of digits raised to the power n
25    while (originalNum != 0) {
26        remainder = originalNum % 10;
27        result += pow(remainder, n);
28        originalNum /= 10;
29    }
30
31    // Check if it's an Armstrong number
32    if (result == num)
33        printf("%d is an Armstrong number.\n", num);
34    else
35        printf("%d is NOT an Armstrong number.\n", num);
36
37    return 0;
38 }
39 /*
40 Enter a number: 371
41 371 is an Armstrong number.
42 Enter a number: 123
43 123 is NOT an Armstrong number.
44 */

```

```

1 //
2 // 6. Write a C program to evaluate the algebraic expression (ax+b)/(ax-b).
3 //
4 #include <stdio.h>
5
6 int main() {
7     float a, x, b, numerator, denominator, result;
8
9     // Input values
10    printf("Enter value for a: ");
11    scanf("%f", &a);
12    printf("Enter value for x: ");
13    scanf("%f", &x);
14    printf("Enter value for b: ");
15    scanf("%f", &b);
16
17    // Calculate numerator and denominator
18    numerator = a * x + b;
19    denominator = a * x - b;
20
21    // Check for division by zero
22    if (denominator == 0) {
23        printf("Error: Denominator becomes zero. Expression is undefined.\n");
24    } else {
25        result = numerator / denominator;
26        printf("The value of (ax + b)/(ax - b) is: %.2f\n", result);
27    }
28
29    return 0;
30 }
31 /*
32 Enter value for a: 2
33 Enter value for x: 3
34 Enter value for b: 4
35 The value of (ax + b)/(ax - b) is: 2.00
36 */

```

```
1 //
2 // 7. Write a C program to check if the given number is perfect number?
3 //
4 #include <stdio.h>
5
6 int main() {
7     int num, sum = 0;
8
9     // Get input from user
10    printf("Enter a number: ");
11    scanf("%d", &num);
12
13    // Find the sum of proper divisors
14    for (int i = 1; i <= num / 2; i++) {
15        if (num % i == 0)
16            sum += i;
17    }
18
19    // Check if sum of divisors equals the number
20    if (sum == num && num != 0)
21        printf("%d is a Perfect Number.\n", num);
22    else
23        printf("%d is NOT a Perfect Number.\n", num);
24
25    return 0;
26 }
27 /*
28 Enter a number: 28
29 28 is a Perfect Number.
30
31 Enter a number: 12
32 12 is NOT a Perfect Number.
33
34 */
```



```
1 //
2 // 8. Write a C program to check if given number is strong number?
3 //
4
5 #include <stdio.h>
6
7 // Function to calculate factorial of a digit
8 int factorial(int n) {
9     int fact = 1;
10    for (int i = 1; i <= n; i++) {
11        fact *= i;
12    }
13    return fact;
14 }
15
16 int main() {
17     int num, originalNum, remainder, sum = 0;
18
19     // Input from user
20     printf("Enter a number: ");
21     scanf("%d", &num);
22
23     originalNum = num;
24
25     // Calculate sum of factorials of digits
26     while (originalNum != 0) {
27         remainder = originalNum % 10;
28         sum += factorial(remainder);
29         originalNum /= 10;
30     }
31
32     // Check if it's a Strong Number
33     if (sum == num)
34         printf("%d is a Strong Number.\n", num);
35     else
36         printf("%d is NOT a Strong Number.\n", num);
37
38     return 0;
39 }
40 /*
41 Enter a number: 145
42 145 is a Strong Number.
43
44 Enter a number: 123
45 123 is NOT a Strong Number.
46 */
```

```
1 //
2 //9. Write a program to print your name without using any semicolon in the program.
3 //
4 #include <stdio.h>
5
6 void main() {
7     if (printf("My name is Ankur.\n")) {}
8
9 }
10
11 /*
12 My name is Ankur.
13 */
```

```

1 //
2 // 10. Write a program to convert temperature in Celsius to Fahrenheit and vice-versa.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int choice;
8     float temp, converted;
9
10    printf("Temperature Conversion Menu:\n");
11    printf("1. Celsius to Fahrenheit\n");
12    printf("2. Fahrenheit to Celsius\n");
13    printf("Enter your choice (1 or 2): ");
14    scanf("%d", &choice);
15
16    if (choice == 1) {
17        printf("Enter temperature in Celsius: ");
18        scanf("%f", &temp);
19        converted = (temp * 9 / 5) + 32;
20        printf("Temperature in Fahrenheit: %.2f F\n", converted);
21    } else if (choice == 2) {
22        printf("Enter temperature in Fahrenheit: ");
23        scanf("%f", &temp);
24        converted = (temp - 32) * 5 / 9;
25        printf("Temperature in Celsius: %.2f C\n", converted);
26    } else {
27        printf("Invalid choice. Please run the program again and choose 1 or 2.\n");
28    }
29
30    return 0;
31 }
32
33 /*
34 Temperature Conversion Menu:
35 1. Celsius to Fahrenheit
36 2. Fahrenheit to Celsius
37 Enter your choice (1 or 2): 1
38 Enter temperature in Celsius: 100
39 Temperature in Fahrenheit: 212.00 F
40 */

```

```
1 //
2 // 11. Write a C program to check whether a number is Palindrome or not.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int num, originalNum, reversed = 0, remainder;
8
9     // Get input from user
10    printf("Enter a number: ");
11    scanf("%d", &num);
12
13    originalNum = num;
14
15    // Reverse the number
16    while (num != 0) {
17        remainder = num % 10;
18        reversed = reversed * 10 + remainder;
19        num /= 10;
20    }
21
22    // Check if original number is equal to reversed
23    if (originalNum == reversed)
24        printf("%d is a Palindrome number.\n", originalNum);
25    else
26        printf("%d is NOT a Palindrome number.\n", originalNum);
27
28    return 0;
29 }
30 /*
31 Enter a number: 1221
32 1221 is a Palindrome number.
33 Enter a number: 1234
34 1234 is NOT a Palindrome number.
35
36 */
```

```
1 //
2 // 12. Write a C program to find maximum between two numbers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int num1, num2;
8
9     // Input two numbers
10    printf("Enter first number: ");
11    scanf("%d", &num1);
12
13    printf("Enter second number: ");
14    scanf("%d", &num2);
15
16    // Compare and display maximum
17    if (num1 > num2)
18        printf("Maximum is: %d\n", num1);
19    else if (num2 > num1)
20        printf("Maximum is: %d\n", num2);
21    else
22        printf("Both numbers are equal.\n");
23
24    return 0;
25 }
26
27 /*
28 Enter first number: 15
29 Enter second number: 30
30 Maximum is: 30
31 */
```

```
1 //
2 // 13. Write a C program to find the maximum between three numbers.
3 //
4
5 #include <stdio.h>
6
7 int main() {
8     int num1, num2, num3, max;
9
10    // Input three numbers
11    printf("Enter three numbers:\n");
12    scanf("%d %d %d", &num1, &num2, &num3);
13
14    // Find maximum using if-else
15    if (num1 >= num2 && num1 >= num3) {
16        max = num1;
17    } else if (num2 >= num1 && num2 >= num3) {
18        max = num2;
19    } else {
20        max = num3;
21    }
22
23    // Output result
24    printf("The maximum number is: %d\n", max);
25
26    return 0;
27 }
28
29 /*
30 Enter three numbers:
31 20 45 38
32 The maximum number is: 45
33 */
34
```

```
1 //
2 // 14. Write a C program to check whether a number is negative, positive or zero.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int num;
8
9     // Input from user
10    printf("Enter a number: ");
11    scanf("%d", &num);
12
13    // Check if the number is positive, negative, or zero
14    if (num > 0)
15        printf("%d is Positive.\n", num);
16    else if (num < 0)
17        printf("%d is Negative.\n", num);
18    else
19        printf("The number is Zero.\n");
20
21    return 0;
22 }
23
24 /*
25 Enter a number: -25
26 -25 is Negative.
27 Enter a number: 0
28 The number is Zero.
29 Enter a number: 18
30 18 is Positive.
31
32 */
```

```
1 //
2 // 15. Write a C program to check whether a number is divisible by 5 and 11 or not within
  the range 100 to 500.
3 //
4 #include <stdio.h>
5
6 int main() {
7     printf("Numbers between 100 and 500 that are divisible by both 5 and 11:\n");
8
9     for (int i = 100; i <= 500; i++) {
10         if (i % 5 == 0 && i % 11 == 0) {
11             printf("%d\n", i);
12         }
13     }
14
15     return 0;
16 }
17 /*
18 Numbers between 100 and 500 that are divisible by both 5 and 11:
19 110
20 165
21 220
22 275
23 330
24 385
25 440
26 495
27
28 */
```



```
1 //
2 // 16. Write a C program to check whether a number is even or odd.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int num;
8
9     // Input from user
10    printf("Enter a number: ");
11    scanf("%d", &num);
12
13    // Check even or odd
14    if (num % 2 == 0)
15        printf("%d is Even.\n", num);
16    else
17        printf("%d is Odd.\n", num);
18
19    return 0;
20 }
21 /*
22 Enter a number: 42
23 42 is Even.
24 Enter a number: 77
25 77 is Odd.
26 */
```

```
1 //
2 // 17. Write a C program to check whether a year is a leap year or not.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int year;
8
9     // Input from user
10    printf("Enter a year: ");
11    scanf("%d", &year);
12
13    // Leap year check logic
14    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
15        printf("%d is a Leap Year.\n", year);
16    } else {
17        printf("%d is NOT a Leap Year.\n", year);
18    }
19
20    return 0;
21 }
22 /*
23 Enter a year: 2020
24 2020 is a Leap Year.
25 Enter a year: 1900
26 1900 is NOT a Leap Year.
27 */
```

```
1 //
2 // 18. Write a C program to check whether a character is alphabet or not.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char ch;
8
9     // Input from user
10    printf("Enter a character: ");
11    scanf(" %c", &ch); // Notice the space before %c to ignore whitespace
12
13    // Check if it's an alphabet
14    if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {
15        printf("'%c' is an alphabet.\n", ch);
16    } else {
17        printf("'%c' is NOT an alphabet.\n", ch);
18    }
19
20    return 0;
21 }
22 /*
23 Enter a character: G
24 'G' is an alphabet.
25
26 Enter a character: 5
27 '5' is NOT an alphabet.
28
29 */
```

```

1 //
2 // 19. Write a C program to input any alphabet and check whether it is vowel or consonant.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char ch;
8
9     // Input character from user
10    printf("Enter an alphabet: ");
11    scanf(" %c", &ch); // Note the space before %c to handle whitespace
12
13    // Check if it's a valid alphabet
14    if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {
15        // Check if it's a vowel
16        if (ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U' ||
17            ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
18            printf("'%c' is a Vowel.\n", ch);
19        } else {
20            printf("'%c' is a Consonant.\n", ch);
21        }
22    } else {
23        printf("'%c' is not a valid alphabet.\n", ch);
24    }
25
26    return 0;
27 }
28 /*
29 Enter an alphabet: e
30 'e' is a Vowel.
31 Enter an alphabet: G
32 'G' is a Consonant.
33 Enter an alphabet: I
34 'I' is not a valid alphabet.
35 */

```

```

1 //
2 // 20. Write a C program to input any character and check whether it is an alphabet, digit or
  special character.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char ch;
8
9     // Input character from user
10    printf("Enter any character: ");
11    scanf(" %c", &ch); // Space before %c to ignore previous newline
12
13    // Check and classify the character
14    if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {
15        printf("'%c' is an Alphabet.\n", ch);
16    } else if (ch >= '0' && ch <= '9') {
17        printf("'%c' is a Digit.\n", ch);
18    } else {
19        printf("'%c' is a Special Character.\n", ch);
20    }
21
22    return 0;
23 }
24 /*
25 Enter any character: M
26 'M' is an Alphabet.
27 Enter any character: 8
28 '8' is a Digit.
29 Enter any character: @
30 '@' is a Special Character.
31 */

```

# Assignment – 2

## Topic: Arrays and Structure

1. Write a C program to store marks for n number of student in an array and print their marks.
2. Write a C 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.
3. Write a C program to insert an element in an array in a particular position.
4. Write a C program to delete an element from a particular position of an array.
5. Write a C program to convert a decimal number taken as input from user to corresponding binary number and store the result in an array.
6. Write a C program to input a binary number in an array and convert into corresponding decimal number.
7. Write a C program to find the smallest and the largest elements in an array.
8. Write a C program for deleting duplicate elements in an array.
9. Write a C program to search for a particular element in an array.
10. Write a C program to sort n elements (ascending order).
11. Write a C program to find second highest number from the array without using sorting.
12. Write a C program to perform addition and subtraction between two matrices.
13. Write a C program to transpose a matrix.
14. Write a C program to add the elements of each row and each column of a matrix.
15. Write a C program to perform the multiplication of two matrices.
16. Write a C program to check whether a matrix is identity matrix or not.
17. Write a C program to check whether a matrix is sparse matrix or not.
18. Write a C program to create a structure named company which has name, address, phone and no of employee as member variables. Read name of company, its address, phone and no of Employee. Finally display these members' value.
19. Define a structure “complex” (typedef) to read two complex numbers and perform addition, subtraction of these two complex numbers and display the result.
20. Write a C program to read Roll No, Name, Address, and Age marks of 12 students in the MCA class and display the details from the function.

```

1 //
2 // 1. Write a C program to store marks for n number of student in an array and print their
  marks.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int n,i;
8
9     // Step 1: Ask user for number of students
10    printf("Enter the number of students: ");
11    scanf("%d", &n);
12
13    // Step 2: Declare an array to store marks
14    float marks[n];
15
16    // Step 3: Input marks for each student
17    for(i = 0; i < n; i++) {
18        printf("Enter marks for student %d: ", i + 1);
19        scanf("%f", &marks[i]);
20    }
21
22    // Step 4: Display the marks
23    printf("\nMarks of students:\n");
24    for(i = 0; i < n; i++) {
25        printf("Student %d: %.2f\n", i + 1, marks[i]);
26    }
27
28    return 0;
29 }
30 /*
31 Enter the number of students: 3
32 Enter marks for student 1: 76.5
33 Enter marks for student 2: 88
34 Enter marks for student 3: 91.25
35
36 Marks of students:
37 Student 1: 76.50
38 Student 2: 88.00
39 Student 3: 91.25
40 */

```

```

1 //
2 // 2. Write a C 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.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int n, i;
8
9     // Step 1: Ask for the number of students
10    printf("Enter the number of students: ");
11    scanf("%d", &n);
12
13    // Step 2: Declare arrays to store marks
14    float math[n], english[n], total[n];
15
16    // Step 3: Input marks for each student
17    for(i = 0; i < n; i++) {
18        printf("\nStudent %d\n", i + 1);
19
20        printf("Enter Mathematics marks: ");
21        scanf("%f", &math[i]);
22
23        printf("Enter English marks: ");
24        scanf("%f", &english[i]);
25
26        // Step 4: Calculate total marks
27        total[i] = math[i] + english[i];
28    }
29
30    // Step 5: Print total marks of each student
31    printf("\nTotal Marks of Each Student:\n");
32    for(i = 0; i < n; i++) {
33        printf("Student %d: %.2f\n", i + 1, total[i]);
34    }
35
36    return 0;
37 }
38
39 /*
40 Enter the number of students: 2
41
42 Student 1
43 Enter Mathematics marks: 78.5
44 Enter English marks: 85
45
46 Student 2
47 Enter Mathematics marks: 90
48 Enter English marks: 88.5
49
50 Total Marks of Each Student:
51 Student 1: 163.50
52 Student 2: 178.50
53 */

```



```

1 //
2 // 3. Write a C program to insert an element in an array in a particular position.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i, pos, element;
8
9     // Step 1: Input the number of elements
10    printf("Enter the number of elements in the array: ");
11    scanf("%d", &n);
12
13    // Step 2: Input the array elements
14    printf("Enter %d elements:\n", n);
15    for(i = 0; i < n; i++) {
16        scanf("%d", &arr[i]);
17    }
18
19    // Step 3: Input the element to insert and the position
20    printf("Enter the element to insert: ");
21    scanf("%d", &element);
22
23    printf("Enter the position (1 to %d) where you want to insert: ", n+1);
24    scanf("%d", &pos);
25
26    // Check for valid position
27    if(pos < 1 || pos > n + 1) {
28        printf("Invalid position!\n");
29        return 1;
30    }
31
32    // Step 4: Shift elements to the right
33    for(i = n; i >= pos; i--) {
34        arr[i] = arr[i - 1];
35    }
36
37    // Step 5: Insert the element
38    arr[pos - 1] = element;
39    n++; // Increase the size
40
41    // Step 6: Print the updated array
42    printf("Array after insertion:\n");
43    for(i = 0; i < n; i++) {
44        printf("%d ", arr[i]);
45    }
46
47    return 0;
48 }
49 /*
50 Enter the number of elements in the array: 5
51 Enter 5 elements:
52 10 20 30 40 50
53 Enter the element to insert: 25
54 Enter the position (1 to 6) where you want to insert: 3
55 Array after insertion:
56 10 20 25 30 40 50
57 */

```

```

1 //
2 // 4. Write a C program to delete an element from a particular position of an array.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i, pos;
8
9     // Step 1: Input the number of elements
10    printf("Enter the number of elements in the array: ");
11    scanf("%d", &n);
12
13    // Step 2: Input array elements
14    printf("Enter %d elements:\n", n);
15    for(i = 0; i < n; i++) {
16        scanf("%d", &arr[i]);
17    }
18
19    // Step 3: Input the position to delete
20    printf("Enter the position (1 to %d) to delete: ", n);
21    scanf("%d", &pos);
22
23    // Step 4: Validate position
24    if(pos < 1 || pos > n) {
25        printf("Invalid position!\n");
26        return 1;
27    }
28
29    // Step 5: Shift elements to the left to overwrite the deleted element
30    for(i = pos - 1; i < n - 1; i++) {
31        arr[i] = arr[i + 1];
32    }
33
34    // Step 6: Reduce size of array
35    n--;
36
37    // Step 7: Print the updated array
38    printf("Array after deletion:\n");
39    for(i = 0; i < n; i++) {
40        printf("%d ", arr[i]);
41    }
42
43    return 0;
44 }
45 /*
46 Enter the number of elements in the array: 5
47 Enter 5 elements:
48 10 20 30 40 50
49 Enter the position (1 to 5) to delete: 3
50 Array after deletion:
51 10 20 40 50
52 */

```

```

1 //
2 // 5. Write a C program to convert a decimal number taken as input from user to
   corresponding binary number and store the result in an array.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int decimal, binary[32], i = 0;
8
9     // Step 1: Input decimal number
10    printf("Enter a decimal number: ");
11    scanf("%d", &decimal);
12
13    // Step 2: Handle special case for 0
14    if (decimal == 0) {
15        binary[i] = 0;
16        i = 1;
17    } else {
18        // Step 3: Convert decimal to binary and store in array
19        while (decimal > 0) {
20            binary[i] = decimal % 2;
21            decimal = decimal / 2;
22            i++;
23        }
24    }
25
26    // Step 4: Print binary number in reverse order
27    printf("Binary equivalent: ");
28    for (int j = i - 1; j >= 0; j--) {
29        printf("%d", binary[j]);
30    }
31
32    printf("\n");
33
34    return 0;
35 }
36 /*
37 Enter a decimal number: 13
38 Binary equivalent: 1101
39 */

```

```

1 //
2 // 6. Write a C program to input a binary number in an array and convert into corresponding
   decimal number.
3 //
4 #include <stdio.h>
5 #include <math.h>
6
7 int main() {
8     int binary[32], n, i, decimal = 0;
9
10    // Step 1: Input the number of binary digits
11    printf("Enter the number of binary digits: ");
12    scanf("%d", &n);
13
14    // Step 2: Input binary digits into the array
15    printf("Enter the binary number (one digit at a time):\n");
16    for(i = 0; i < n; i++) {
17        scanf("%d", &binary[i]);
18
19        // Optional: Validate input
20        if (binary[i] != 0 && binary[i] != 1) {
21            printf("Invalid binary digit! Please enter only 0 or 1.\n");
22            return 1;
23        }
24    }
25
26    // Step 3: Convert binary to decimal
27    for(i = 0; i < n; i++) {
28        decimal += binary[i] * pow(2, n - 1 - i);
29    }
30
31    // Step 4: Print the decimal number
32    printf("Decimal equivalent: %d\n", decimal);
33
34    return 0;
35 }
36 /*
37 Enter the number of binary digits: 4
38 Enter the binary number (one digit at a time):
39 1
40 1
41 0
42 1
43 Decimal equivalent: 13
44 */

```

```

1 //
2 // 7. Write a C program to find the smallest and the largest elements in an array.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i;
8     int smallest, largest;
9
10    // Step 1: Input the number of elements
11    printf("Enter the number of elements in the array: ");
12    scanf("%d", &n);
13
14    // Step 2: Input array elements
15    printf("Enter %d elements:\n", n);
16    for(i = 0; i < n; i++) {
17        scanf("%d", &arr[i]);
18    }
19
20    // Step 3: Initialize smallest and largest with first element
21    smallest = largest = arr[0];
22
23    // Step 4: Traverse the array to find smallest and largest
24    for(i = 1; i < n; i++) {
25        if(arr[i] < smallest)
26            smallest = arr[i];
27        if(arr[i] > largest)
28            largest = arr[i];
29    }
30
31    // Step 5: Print results
32    printf("Smallest element: %d\n", smallest);
33    printf("Largest element: %d\n", largest);
34
35    return 0;
36 }
37 /*
38 Enter the number of elements in the array: 5
39 Enter 5 elements:
40 25 10 65 3 90
41 Smallest element: 3
42 Largest element: 90
43 */

```

```

1 //
2 // 8. Write a C program for deleting duplicate elements in an array.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i, j, k;
8
9     // Step 1: Input array size
10    printf("Enter the number of elements in the array: ");
11    scanf("%d", &n);
12
13    // Step 2: Input array elements
14    printf("Enter %d elements:\n", n);
15    for(i = 0; i < n; i++) {
16        scanf("%d", &arr[i]);
17    }
18
19    // Step 3: Remove duplicates
20    for(i = 0; i < n; i++) {
21        for(j = i + 1; j < n; ) {
22            if(arr[i] == arr[j]) {
23                // Shift elements left to remove duplicate
24                for(k = j; k < n - 1; k++) {
25                    arr[k] = arr[k + 1];
26                }
27                n--; // Reduce array size
28            } else {
29                j++; // Only move to next if no deletion
30            }
31        }
32    }
33
34    // Step 4: Print the updated array
35    printf("Array after deleting duplicates:\n");
36    for(i = 0; i < n; i++) {
37        printf("%d ", arr[i]);
38    }
39
40    return 0;
41 }
42 /*
43 Enter the number of elements in the array: 7
44 Enter 7 elements:
45 1 2 3 2 4 1 5
46 Array after deleting duplicates:
47 1 2 3 4 5
48 */

```

```

1 //
2 // 9. Write a C program to search for a particular element in an array.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i, search, found = 0;
8
9     // Step 1: Input array size
10    printf("Enter the number of elements in the array: ");
11    scanf("%d", &n);
12
13    // Step 2: Input array elements
14    printf("Enter %d elements:\n", n);
15    for(i = 0; i < n; i++) {
16        scanf("%d", &arr[i]);
17    }
18
19    // Step 3: Input the element to search
20    printf("Enter the element to search: ");
21    scanf("%d", &search);
22
23    // Step 4: Search for the element (linear search)
24    for(i = 0; i < n; i++) {
25        if(arr[i] == search) {
26            found = 1;
27            break;
28        }
29    }
30
31    // Step 5: Print result
32    if(found) {
33        printf("Element %d found at position %d (index %d).\n", search, i + 1, i);
34    } else {
35        printf("Element %d not found in the array.\n", search);
36    }
37
38    return 0;
39 }
40 /*
41 Enter the number of elements in the array: 5
42 Enter 5 elements:
43 10 20 30 40 50
44 Enter the element to search: 30
45 Element 30 found at position 3 (index 2).
46 */

```

```

1 //
2 // 10. Write a C program to sort n elements (ascending order).
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i, j, temp;
8
9     // Step 1: Input number of elements
10    printf("Enter the number of elements: ");
11    scanf("%d", &n);
12
13    // Step 2: Input array elements
14    printf("Enter %d elements:\n", n);
15    for(i = 0; i < n; i++) {
16        scanf("%d", &arr[i]);
17    }
18
19    // Step 3: Bubble Sort to sort the array in ascending order
20    for(i = 0; i < n - 1; i++) {
21        for(j = 0; j < n - i - 1; j++) {
22            if(arr[j] > arr[j + 1]) {
23                // Swap arr[j] and arr[j + 1]
24                temp = arr[j];
25                arr[j] = arr[j + 1];
26                arr[j + 1] = temp;
27            }
28        }
29    }
30
31    // Step 4: Print the sorted array
32    printf("Sorted array in ascending order:\n");
33    for(i = 0; i < n; i++) {
34        printf("%d ", arr[i]);
35    }
36
37    return 0;
38 }
39 /*
40 Enter the number of elements: 5
41 Enter 5 elements:
42 45 10 30 20 25
43 Sorted array in ascending order:
44 10 20 25 30 45
45 */

```



```

1 //
2 // 11. Write a C program to find second highest number from the array without using sorting
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, i;
8     int first, second;
9
10    // Step 1: Input number of elements
11    printf("Enter the number of elements: ");
12    scanf("%d", &n);
13
14    // Check for minimum size
15    if(n < 2) {
16        printf("At least two elements are required.\n");
17        return 1;
18    }
19
20    // Step 2: Input array elements
21    printf("Enter %d elements:\n", n);
22    for(i = 0; i < n; i++) {
23        scanf("%d", &arr[i]);
24    }
25
26    // Step 3: Initialize first and second highest
27    first = second = -2147483648; // Smallest possible integer
28
29    // Step 4: Traverse to find first and second highest
30    for(i = 0; i < n; i++) {
31        if(arr[i] > first) {
32            second = first;
33            first = arr[i];
34        } else if(arr[i] > second && arr[i] < first) {
35            second = arr[i];
36        }
37    }
38
39    // Step 5: Check and print result
40    if(second == -2147483648) {
41        printf("There is no second highest element (all elements might be same).\n");
42    } else {
43        printf("The second highest element is: %d\n", second);
44    }
45
46    return 0;
47 }
48 /*
49 Enter the number of elements: 6
50 Enter 6 elements:
51 10 45 32 67 67 23
52 The second highest element is: 45
53 */

```

```

1 //
2 // 12. Write a C program to perform addition and subtraction between two matrices.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int A[10][10], B[10][10], sum[10][10], diff[10][10];
8     int rows, cols, i, j;
9
10    // Step 1: Input dimensions
11    printf("Enter number of rows and columns of the matrices: ");
12    scanf("%d %d", &rows, &cols);
13
14    // Step 2: Input first matrix
15    printf("Enter elements of first matrix (A):\n");
16    for(i = 0; i < rows; i++) {
17        for(j = 0; j < cols; j++) {
18            scanf("%d", &A[i][j]);
19        }
20    }
21
22    // Step 3: Input second matrix
23    printf("Enter elements of second matrix (B):\n");
24    for(i = 0; i < rows; i++) {
25        for(j = 0; j < cols; j++) {
26            scanf("%d", &B[i][j]);
27        }
28    }
29
30    // Step 4: Perform addition and subtraction
31    for(i = 0; i < rows; i++) {
32        for(j = 0; j < cols; j++) {
33            sum[i][j] = A[i][j] + B[i][j];
34            diff[i][j] = A[i][j] - B[i][j];
35        }
36    }
37
38    // Step 5: Display results
39    printf("\nSum of the matrices (A + B):\n");
40    for(i = 0; i < rows; i++) {
41        for(j = 0; j < cols; j++) {
42            printf("%d ", sum[i][j]);
43        }
44        printf("\n");
45    }
46
47    printf("\nDifference of the matrices (A - B):\n");
48    for(i = 0; i < rows; i++) {
49        for(j = 0; j < cols; j++) {
50            printf("%d ", diff[i][j]);
51        }
52        printf("\n");
53    }
54
55    return 0;
56 }
57 /*

```

58 *Enter number of rows and columns of the matrices: 2 2*

59 *Enter elements of first matrix (A):*

60 *1 2*

61 *3 4*

62 *Enter elements of second matrix (B):*

63 *5 6*

64 *7 8*

65

66 *Sum of the matrices (A + B):*

67 *6 8*

68 *10 12*

69

70 *Difference of the matrices (A - B):*

71 *-4 -4*

72 *-4 -4*

73 *\*/*

```

1 //
2 // 13. Write a C program to transpose a matrix.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int matrix[10][10], transpose[10][10];
8     int rows, cols, i, j;
9
10    // Step 1: Input matrix size
11    printf("Enter the number of rows and columns of the matrix: ");
12    scanf("%d %d", &rows, &cols);
13
14    // Step 2: Input matrix elements
15    printf("Enter the elements of the matrix:\n");
16    for(i = 0; i < rows; i++) {
17        for(j = 0; j < cols; j++) {
18            scanf("%d", &matrix[i][j]);
19        }
20    }
21
22    // Step 3: Transpose the matrix
23    for(i = 0; i < rows; i++) {
24        for(j = 0; j < cols; j++) {
25            transpose[j][i] = matrix[i][j];
26        }
27    }
28
29    // Step 4: Print the transposed matrix
30    printf("\nTranspose of the matrix:\n");
31    for(i = 0; i < cols; i++) {
32        for(j = 0; j < rows; j++) {
33            printf("%d ", transpose[i][j]);
34        }
35        printf("\n");
36    }
37
38    return 0;
39 }
40 /*
41 Enter the number of rows and columns of the matrix: 2 3
42 Enter the elements of the matrix:
43 1 2 3
44 4 5 6
45
46 Transpose of the matrix:
47 1 4
48 2 5
49 3 6
50 */

```

```

1 //
2 // 14. Write a C program to add the elements of each row and each column of a matrix.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int matrix[10][10];
8     int rows, cols, i, j;
9     int rowSum, colSum;
10
11     // Step 1: Input matrix size
12     printf("Enter number of rows and columns: ");
13     scanf("%d %d", &rows, &cols);
14
15     // Step 2: Input matrix elements
16     printf("Enter elements of the matrix:\n");
17     for(i = 0; i < rows; i++) {
18         for(j = 0; j < cols; j++) {
19             scanf("%d", &matrix[i][j]);
20         }
21     }
22
23     // Step 3: Calculate and display row sums
24     printf("\nSum of each row:\n");
25     for(i = 0; i < rows; i++) {
26         rowSum = 0;
27         for(j = 0; j < cols; j++) {
28             rowSum += matrix[i][j];
29         }
30         printf("Row %d = %d\n", i + 1, rowSum);
31     }
32
33     // Step 4: Calculate and display column sums
34     printf("\nSum of each column:\n");
35     for(j = 0; j < cols; j++) {
36         colSum = 0;
37         for(i = 0; i < rows; i++) {
38             colSum += matrix[i][j];
39         }
40         printf("Column %d = %d\n", j + 1, colSum);
41     }
42
43     return 0;
44 }
45 /*
46 Enter number of rows and columns: 2 3
47 Enter elements of the matrix:
48 1 2 3
49 4 5 6
50
51 Sum of each row:
52 Row 1 = 6
53 Row 2 = 15
54
55 Sum of each column:
56 Column 1 = 5
57 Column 2 = 7

```

58 *Column 3 = 9*

59 \*/

```

1 //
2 // 15. Write a C program to perform the multiplication of two matrices.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int A[10][10], B[10][10], C[10][10];
8     int r1, c1, r2, c2, i, j, k;
9
10    // Step 1: Input dimensions
11    printf("Enter rows and columns of first matrix (A): ");
12    scanf("%d %d", &r1, &c1);
13
14    printf("Enter rows and columns of second matrix (B): ");
15    scanf("%d %d", &r2, &c2);
16
17    // Step 2: Check if multiplication is possible
18    if(c1 != r2) {
19        printf("Matrix multiplication not possible! Columns of A must equal rows of B.\n");
20        return 1;
21    }
22
23    // Step 3: Input matrix A
24    printf("Enter elements of matrix A (%d x %d):\n", r1, c1);
25    for(i = 0; i < r1; i++) {
26        for(j = 0; j < c1; j++) {
27            scanf("%d", &A[i][j]);
28        }
29    }
30
31    // Step 4: Input matrix B
32    printf("Enter elements of matrix B (%d x %d):\n", r2, c2);
33    for(i = 0; i < r2; i++) {
34        for(j = 0; j < c2; j++) {
35            scanf("%d", &B[i][j]);
36        }
37    }
38
39    // Step 5: Initialize result matrix C to 0
40    for(i = 0; i < r1; i++) {
41        for(j = 0; j < c2; j++) {
42            C[i][j] = 0;
43        }
44    }
45
46    // Step 6: Perform matrix multiplication
47    for(i = 0; i < r1; i++) {
48        for(j = 0; j < c2; j++) {
49            for(k = 0; k < c1; k++) {
50                C[i][j] += A[i][k] * B[k][j];
51            }
52        }
53    }
54
55    // Step 7: Display result
56    printf("\nResultant Matrix (A x B):\n");
57    for(i = 0; i < r1; i++) {

```

```
58     for(j = 0; j < c2; j++) {
59         printf("%d ", C[i][j]);
60     }
61     printf("\n");
62 }
63
64 return 0;
65 }
66 /*
67  Enter rows and columns of first matrix (A): 2 3
68  Enter rows and columns of second matrix (B): 3 2
69  Enter elements of matrix A (2 x 3):
70  1 2 3
71  4 5 6
72  Enter elements of matrix B (3 x 2):
73  7 8
74  9 10
75  11 12
76
77  Resultant Matrix (A x B):
78  58 64
79  139 154
80  */
```



```

1 //
2 // 16. Write a C program to check whether a matrix is identity matrix or not.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int matrix[10][10];
8     int i, j, size, isIdentity = 1;
9
10    // Step 1: Input matrix size (square)
11    printf("Enter the size of the square matrix (n x n): ");
12    scanf("%d", &size);
13
14    // Step 2: Input matrix elements
15    printf("Enter the elements of the %dx%d matrix:\n", size, size);
16    for(i = 0; i < size; i++) {
17        for(j = 0; j < size; j++) {
18            scanf("%d", &matrix[i][j]);
19        }
20    }
21
22    // Step 3: Check identity matrix condition
23    for(i = 0; i < size; i++) {
24        for(j = 0; j < size; j++) {
25            if(i == j && matrix[i][j] != 1) {
26                isIdentity = 0; // Diagonal element is not 1
27            } else if(i != j && matrix[i][j] != 0) {
28                isIdentity = 0; // Non-diagonal element is not 0
29            }
30        }
31    }
32
33    // Step 4: Display result
34    if(isIdentity) {
35        printf("The matrix is an Identity Matrix.\n");
36    } else {
37        printf("The matrix is NOT an Identity Matrix.\n");
38    }
39
40    return 0;
41 }
42 /*
43 Enter the size of the square matrix (n x n): 3
44 Enter the elements of the 3x3 matrix:
45 1 0 0
46 0 1 0
47 0 0 1
48 The matrix is an Identity Matrix.
49 */

```

```

1 //
2 // 17. Write a C program to check whether a matrix is sparse matrix or not.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int matrix[10][10];
8     int rows, cols, i, j;
9     int zeroCount = 0, totalElements;
10
11     // Step 1: Input matrix size
12     printf("Enter the number of rows and columns of the matrix: ");
13     scanf("%d %d", &rows, &cols);
14
15     totalElements = rows * cols;
16
17     // Step 2: Input matrix elements
18     printf("Enter elements of the matrix:\n");
19     for(i = 0; i < rows; i++) {
20         for(j = 0; j < cols; j++) {
21             scanf("%d", &matrix[i][j]);
22             if(matrix[i][j] == 0) {
23                 zeroCount++;
24             }
25         }
26     }
27
28     // Step 3: Check if matrix is sparse
29     if(zeroCount > totalElements / 2) {
30         printf("The matrix is a Sparse Matrix.\n");
31     } else {
32         printf("The matrix is NOT a Sparse Matrix.\n");
33     }
34
35     return 0;
36 }
37 /*
38 Enter the number of rows and columns of the matrix: 3 3
39 Enter elements of the matrix:
40 0 0 1
41 0 0 0
42 0 2 0
43 The matrix is a Sparse Matrix.
44 */

```

```

1 //
2 // 18. Write a C program to create a structure named company which has name, address,
  phone and no of employee as member variables. Read name of company, its address, phone
  and no of Employee. Finally display these members' value.
3 //
4 #include <stdio.h>
5
6 // Define the structure
7 struct company {
8     char name[100];
9     char address[200];
10    char phone[20];
11    int noOfEmployees;
12 };
13
14 int main() {
15     struct company c;
16
17     // Input company details
18     printf("Enter company name: ");
19     fgets(c.name, sizeof(c.name), stdin);
20
21     printf("Enter company address: ");
22     fgets(c.address, sizeof(c.address), stdin);
23
24     printf("Enter company phone: ");
25     fgets(c.phone, sizeof(c.phone), stdin);
26
27     printf("Enter number of employees: ");
28     scanf("%d", &c.noOfEmployees);
29
30     // Display company details
31     printf("\n--- Company Details ---\n");
32     printf("Name: %s", c.name);
33     printf("Address: %s", c.address);
34     printf("Phone: %s", c.phone);
35     printf("Number of Employees: %d\n", c.noOfEmployees);
36
37     return 0;
38 }
39 /*
40 Enter company name: Tech Innovators
41 Enter company address: 123 Silicon Valley, Newtown
42 Enter company phone: 123-456-7890
43 Enter number of employees: 250
44
45 --- Company Details ---
46 Name: Tech Innovators
47 Address: 123 Silicon Valley, Newtown
48 Phone: 123-456-7890
49 Number of Employees: 250
50 */

```

```

1 //
2 // 19. Define a structure "complex" (typedef) to read two complex numbers and perform
   addition, subtraction of these two complex numbers and display the result.
3 //
4 #include <stdio.h>
5
6 // Define the structure using typedef
7 typedef struct {
8     float real;
9     float imag;
10 } Complex;
11
12 // Function to add two complex numbers
13 Complex add(Complex a, Complex b) {
14     Complex result;
15     result.real = a.real + b.real;
16     result.imag = a.imag + b.imag;
17     return result;
18 }
19
20 // Function to subtract two complex numbers
21 Complex subtract(Complex a, Complex b) {
22     Complex result;
23     result.real = a.real - b.real;
24     result.imag = a.imag - b.imag;
25     return result;
26 }
27
28 // Main program
29 int main() {
30     Complex c1, c2, sum, diff;
31
32     // Read first complex number
33     printf("Enter real and imaginary part of first complex number: ");
34     scanf("%f %f", &c1.real, &c1.imag);
35
36     // Read second complex number
37     printf("Enter real and imaginary part of second complex number: ");
38     scanf("%f %f", &c2.real, &c2.imag);
39
40     // Perform operations
41     sum = add(c1, c2);
42     diff = subtract(c1, c2);
43
44     // Display results
45     printf("\nSum = %.2f + %.2fi\n", sum.real, sum.imag);
46     printf("Difference = %.2f + %.2fi\n", diff.real, diff.imag);
47
48     return 0;
49 }
50 /*
51 Enter real and imaginary part of first complex number: 3 4
52 Enter real and imaginary part of second complex number: 1 2
53
54 Sum = 4.00 + 6.00i
55 Difference = 2.00 + 2.00i
56 */

```

```

1 //
2 // 20. Write a C program to read Roll No, Name, Address, and Age marks of 12 students in
  the MCA class and display the details from the function.
3 //
4 #include <stdio.h>
5
6 #define SIZE 12
7
8 // Define structure for student
9 struct Student {
10     int rollNo;
11     char name[100];
12     char address[200];
13     int age;
14     float marks;
15 };
16
17 // Function to display student data
18 void displayStudents(struct Student s[], int size) {
19     printf("\n--- MCA Students Record ---\n");
20     for(int i = 0; i < size; i++) {
21         printf("\nStudent %d\n", i + 1);
22         printf("Roll No : %d\n", s[i].rollNo);
23         printf("Name : %s", s[i].name);
24         printf("Address : %s", s[i].address);
25         printf("Age : %d\n", s[i].age);
26         printf("Marks : %.2f\n", s[i].marks);
27     }
28 }
29
30 int main() {
31     struct Student students[SIZE];
32
33     // Read student data
34     for(int i = 0; i < SIZE; i++) {
35         printf("\nEnter details for Student %d\n", i + 1);
36
37         printf("Roll No: ");
38         scanf("%d", &students[i].rollNo);
39         getchar(); // consume newline
40
41         printf("Name: ");
42         fgets(students[i].name, sizeof(students[i].name), stdin);
43
44         printf("Address: ");
45         fgets(students[i].address, sizeof(students[i].address), stdin);
46
47         printf("Age: ");
48         scanf("%d", &students[i].age);
49
50         printf("Marks: ");
51         scanf("%f", &students[i].marks);
52         getchar(); // consume newline
53     }
54
55     // Call display function
56     displayStudents(students, SIZE);

```

```
57
58     return 0;
59 }
60
61 /*
62  Enter details for Student 1
63  Roll No: 101
64  Name: Anjali Sharma
65  Address: 45 Green Park, Delhi
66  Age: 22
67  Marks: 88.5
68
69  Enter details for Student 2
70  Roll No: 102
71  Name: Rohan Mehta
72  Address: 12 Nehru Road, Mumbai
73  Age: 23
74  Marks: 79.0
75
76  --- MCA Students Record ---
77
78  Student 1
79  Roll No : 101
80  Name   : Anjali Sharma
81  Address : 45 Green Park, Delhi
82  Age    : 22
83  Marks  : 88.50
84
85  Student 2
86  Roll No : 102
87  Name    : Rohan Mehta
88  Address : 12 Nehru Road, Mumbai
89  Age     : 23
90  Marks   : 79.00
91  */
```

# Assignment – 3

## Topic: Functions

1. Write a C program to add, subtract, multiply and divide two integers using a user-defined type function with return type.
2. Write a C program to calculate sum of first 20 natural numbers using recursive function.
3. Write a C program to generate Fibonacci series using recursive function.
4. Write a C program to swap two integers using call by value and call by reference methods of passing arguments to a function.
5. Write a C program to find sum of digits of the number using Recursive Function.
6. Write a C program to read an integer number and print the reverse of that number using recursion.
7. Write a C program to find maximum and minimum between two numbers using functions.
8. Write a C program to check whether a number is even or odd using functions.
9. Write a C program to check whether a number is prime, Armstrong or perfect number using functions.
10. Write a C program to find power of any number using recursion.

```
1 //
2 //1. Write a C program to add, subtract, multiply and divide two integers using a user-
  defined type function with return type.
3 //
4
5 #ifndef OPERATIONS_H
6 #define OPERATIONS_H
7 int add(int a, int b);
8 int subtract(int a, int b);
9 int multiply(int a, int b);
10 float divide(int a, int b);
11 #endif //OPERATIONS_H
12
```



```
1 //
2 //1. Write a C program to add, subtract, multiply and divide two integers using a user-
  defined type function with return type.
3 //
4
5 #include "operations.h"
6
7 int add(int a, int b) {
8     return a + b;
9 }
10
11 int subtract(int a, int b) {
12     return a - b;
13 }
14
15 int multiply(int a, int b) {
16     return a * b;
17 }
18
19 float divide(int a, int b) {
20     if (b == 0) {
21         return 0.0; // basic error handling
22     }
23     return (float)a / b;
24 }
```

```

1 //
2 // 1. Write a C program to add, subtract, multiply and divide two integers using a user-
   defined type function with return type.
3 //
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include "operations.h"
7
8 int main() {
9     int choice, num1, num2;
10    float result;
11
12    while (1) {
13        printf("\n==== Calculator Menu ==== \n");
14        printf("1. Addition\n");
15        printf("2. Subtraction\n");
16        printf("3. Multiplication\n");
17        printf("4. Division\n");
18        printf("5. Exit\n");
19        printf("Choose an option: ");
20        scanf("%d", &choice);
21
22        if (choice == 5) {
23            printf("Exiting program. Goodbye!\n");
24            break;
25        }
26
27        printf("Enter two integers: ");
28        scanf("%d %d", &num1, &num2);
29
30        switch (choice) {
31            case 1:
32                printf("Result: %d\n", add(num1, num2));
33                break;
34            case 2:
35                printf("Result: %d\n", subtract(num1, num2));
36                break;
37            case 3:
38                printf("Result: %d\n", multiply(num1, num2));
39                break;
40            case 4:
41                if (num2 == 0)
42                    printf("Error: Division by zero not allowed.\n");
43                else {
44                    result = divide(num1, num2);
45                    printf("Result: %.2f\n", result);
46                }
47                break;
48            default:
49                printf("Invalid choice! Please try again.\n");
50        }
51    }
52
53    return 0;
54 }
55 /*
56 gcc A3Q1-Calculator.c operations.c -o calculator

```

```
57 ./calculator
58
59 ===== Calculator Menu =====
60 1. Addition
61 2. Subtraction
62 3. Multiplication
63 4. Division
64 5. Exit
65 Choose an option: 1
66 Enter two integers: 34
67 55
68 Result: 89
69
70 ===== Calculator Menu =====
71 1. Addition
72 2. Subtraction
73 3. Multiplication
74 4. Division
75 5. Exit
76 Choose an option: 4
77 Enter two integers: 34 7
78 Result: 4.86
79
80 ===== Calculator Menu =====
81 1. Addition
82 2. Subtraction
83 3. Multiplication
84 4. Division
85 5. Exit
86 Choose an option: 5
87 Exiting program. Goodbye!
88 */
```

```
1 //
2 //2. Write a C program to calculate sum of first 20 natural numbers using recursive function.
3 //
4 #include <stdio.h>
5
6 // Recursive function to calculate sum
7 int sum(int n) {
8     if (n == 1)
9         return 1;
10    else
11        return n + sum(n - 1);
12 }
13
14 int main() {
15     int result = sum(20);
16     printf("Sum of the first 20 natural numbers is: %d\n", result);
17     return 0;
18 }
19
20 /*
21 Sum of the first 20 natural numbers is: 210
22 */
```

```
1 //
2 // 3. Write a C program to generate Fibonacci series using recursive function.
3 //
4 #include <stdio.h>
5
6 // Recursive function to calculate Fibonacci number
7 int fibonacci(int n) {
8     if (n == 0)
9         return 0;
10    else if (n == 1)
11        return 1;
12    else
13        return fibonacci(n - 1) + fibonacci(n - 2);
14 }
15
16 int main() {
17     int i, terms;
18
19     printf("Enter the number of terms in Fibonacci series: ");
20     scanf("%d", &terms);
21
22     printf("Fibonacci series up to %d terms:\n", terms);
23     for (i = 0; i < terms; i++) {
24         printf("%d ", fibonacci(i));
25     }
26
27     printf("\n");
28     return 0;
29 }
30
31 /*
32 Enter the number of terms in Fibonacci series: 7
33
34 Fibonacci series up to 7 terms:
35 0 1 1 2 3 5 8
36 */
```

```
1 #include <stdio.h>
2
3 // Call by Value
4 void swapByValue(int a, int b) {
5     int temp = a;
6     a = b;
7     b = temp;
8     printf("Inside swapByValue - a: %d, b: %d\n", a, b);
9 }
10
11 // Call by Reference
12 void swapByReference(int *a, int *b) {
13     int temp = *a;
14     *a = *b;
15     *b = temp;
16 }
17
18
```

```
1 //
2 // 4. Write a C program to swap two integers using call by value and call by reference
  methods of passing arguments to a function.
3 //
4 #include <stdio.h>
5 #include "A3Q4 - Swap.h"
6 int main() {
7     int x = 5, y = 10;
8
9     printf("Before swapByValue - x: %d, y: %d\n", x, y);
10    swapByValue(x, y);
11    printf("After swapByValue - x: %d, y: %d\n", x, y);
12
13    printf("\nBefore swapByReference - x: %d, y: %d\n", x, y);
14    swapByReference(&x, &y);
15    printf("After swapByReference - x: %d, y: %d\n", x, y);
16
17    return 0;
18 }
19 /*
20 Before swapByValue - x: 5, y: 10
21 Inside swapByValue - a: 10, b: 5
22 After swapByValue - x: 5, y: 10
23
24 Before swapByReference - x: 5, y: 10
25 After swapByReference - x: 10, y: 5
26 */
```

```
1 //
2 // 5. Write a C program to find sum of digits of the number using Recursive Function.
3 //
4 #include <stdio.h>
5
6 // Recursive function to calculate sum of digits
7 int sumOfDigits(int num) {
8     if (num == 0)
9         return 0;
10    else
11        return (num % 10) + sumOfDigits(num / 10);
12 }
13
14 int main() {
15     int number;
16
17     // Taking input from user
18     printf("Enter a number: ");
19     scanf("%d", &number);
20
21     // Handling negative input
22     if (number < 0) {
23         number = -number;
24     }
25
26     // Calling the recursive function and displaying the result
27     int result = sumOfDigits(number);
28     printf("Sum of digits: %d\n", result);
29
30     return 0;
31 }
32 /*
33 Enter a number: 1234
34 Sum of digits: 10
35 */
```



```
1 //
2 // 6. Write a C program to read an integer number and print the reverse of that number using
  recursion.
3 //
4 #include <stdio.h>
5
6 // Function prototype
7 void reverseNumber(int num);
8
9 int main() {
10     int number;
11
12     // Input from user
13     printf("Enter an integer: ");
14     scanf("%d", &number);
15
16     // Handle negative number
17     if (number < 0) {
18         printf("-");
19         number = -number;
20     }
21
22     printf("Reversed number: ");
23     reverseNumber(number);
24     printf("\n");
25
26     return 0;
27 }
28
29 // Recursive function to print digits in reverse order
30 void reverseNumber(int num) {
31     if (num == 0)
32         return;
33
34     printf("%d", num % 10);
35     reverseNumber(num / 10);
36 }
37
38 /*
39 Enter an integer: 12345
40 Reversed number: 54321
41 */
```

```
1 //
2 // 7. Write a C program to find maximum and minimum between two numbers using functions.
3 //
4 #include <stdio.h>
5
6 // Function to find maximum
7 int findMax(int a, int b) {
8     return (a > b) ? a : b;
9 }
10
11 // Function to find minimum
12 int findMin(int a, int b) {
13     return (a < b) ? a : b;
14 }
15
16 int main() {
17     int num1, num2;
18
19     // Input from user
20     printf("Enter two numbers: ");
21     scanf("%d %d", &num1, &num2);
22
23     // Calling functions and displaying results
24     printf("Maximum: %d\n", findMax(num1, num2));
25     printf("Minimum: %d\n", findMin(num1, num2));
26
27     return 0;
28 }
29
30 /*
31 Enter two numbers: 15 27
32 Maximum: 27
33 Minimum: 15
34 */
```

```
1 //
2 // 8. Write a C program to check whether a number is even or odd using functions.
3 //
4 #include <stdio.h>
5
6 // Function to check even or odd
7 void checkEvenOdd(int num) {
8     if (num % 2 == 0)
9         printf("%d is Even.\n", num);
10    else
11        printf("%d is Odd.\n", num);
12 }
13
14 int main() {
15     int number;
16
17     // Input from user
18     printf("Enter a number: ");
19     scanf("%d", &number);
20
21     // Function call
22     checkEvenOdd(number);
23
24     return 0;
25 }
26 /*
27 Enter a number: 42
28 42 is Even.
29 Enter a number: 31
30 31 is Odd.
31 */
```

```

1 //
2 //9. Write a C program to check whether a number is prime, Armstrong or perfect number
  using functions.
3 //
4 #include <stdio.h>    // For input/output functions
5 #include <math.h>     // For power() function (used in Armstrong check)
6 #include <stdbool.h>  // For boolean type (true/false)
7
8 // Function to check Prime number
9 bool isPrime(int num) {
10     if (num <= 1)
11         return false;
12
13     for (int i = 2; i <= sqrt(num); i++) {
14         if (num % i == 0)
15             return false;
16     }
17     return true;
18 }
19
20 // Function to check Armstrong number
21 bool isArmstrong(int num) {
22     int original = num, sum = 0, digits = 0;
23
24     // Count the number of digits
25     int temp = num;
26     while (temp != 0) {
27         digits++;
28         temp /= 10;
29     }
30
31     temp = num;
32     while (temp != 0) {
33         int digit = temp % 10;
34         sum += pow(digit, digits); // Use power function from math.h
35         temp /= 10;
36     }
37
38     return (sum == original);
39 }
40
41 // Function to check Perfect number
42 bool isPerfect(int num) {
43     if (num <= 0) return false;
44
45     int sum = 0;
46     for (int i = 1; i < num; i++) {
47         if (num % i == 0)
48             sum += i;
49     }
50
51     return (sum == num);
52 }
53
54 int main() {
55     int number;
56

```

```
57 // Input from user
58 printf("Enter a number: ");
59 scanf("%d", &number);
60
61 // Prime check
62 if (isPrime(number))
63     printf("%d is a Prime number.\n", number);
64 else
65     printf("%d is not a Prime number.\n", number);
66
67 // Armstrong check
68 if (isArmstrong(number))
69     printf("%d is an Armstrong number.\n", number);
70 else
71     printf("%d is not an Armstrong number.\n", number);
72
73 // Perfect check
74 if (isPerfect(number))
75     printf("%d is a Perfect number.\n", number);
76 else
77     printf("%d is not a Perfect number.\n", number);
78
79 return 0;
80 }
81 /*
82 Enter a number: 153
83 153 is not a Prime number.
84 153 is an Armstrong number.
85 153 is not a Perfect number.
86 */
```

```

1 //
2 //10. Write a C program to find power of any number using recursion.
3 //
4 #include <stdio.h>
5
6 // Recursive function to calculate power
7 int power(int base, int exponent) {
8     if (exponent == 0)
9         return 1;
10    else
11        return base * power(base, exponent - 1);
12 }
13
14 int main() {
15     int base, exponent;
16
17     // Input from user
18     printf("Enter base: ");
19     scanf("%d", &base);
20
21     printf("Enter exponent: ");
22     scanf("%d", &exponent);
23
24     // Handling negative exponents (optional)
25     if (exponent < 0) {
26         printf("This program does not support negative exponents.\n");
27     } else {
28         int result = power(base, exponent);
29         printf("%d^%d = %d\n", base, exponent, result);
30     }
31
32     return 0;
33 }
34 /*
35 Enter base: 2
36 Enter exponent: 5
37 2^5 = 32
38 */

```

# Assignment – 4

## Topic: Pointers

1. Write a C program to find the sum of all the elements of an array using pointers.
2. Write a C program to swap value of two variables using pointer.
3. Write a C program to add two numbers using pointers.
4. Write a C program to input and print array elements using pointer.
5. Write a C program to copy one array to another using pointer.
6. Write a C program to swap two arrays using pointers.
7. Write a C program to reverse an array using pointers.
8. Write a C program to search for an element in array using pointers.
9. Write a C program to add two 2 X 2 matrix using pointers.
10. Write a C program to multiply two 2 X 2 matrix using pointers.
11. Write a C program to find length of string using pointers.
12. Write a C program to copy one string to another using pointer.
13. Write a C program to concatenate two strings using pointers.
14. Write a C program to compare two strings using pointers.
15. Write a C program to find a substring from a given string using pointers.

```
1 //
2 // 1. Write a C program to find the sum of all the elements of an array using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, sum = 0;
8     int *ptr;
9
10    // Input array size
11    printf("Enter the number of elements in the array: ");
12    scanf("%d", &n);
13
14    // Input array elements
15    printf("Enter %d elements:\n", n);
16    for (int i = 0; i < n; i++) {
17        scanf("%d", &arr[i]);
18    }
19
20    // Pointer pointing to the start of the array
21    ptr = arr;
22
23    // Sum using pointer
24    for (int i = 0; i < n; i++) {
25        sum += *(ptr + i);
26    }
27
28    printf("Sum of all elements = %d\n", sum);
29
30    return 0;
31 }
32 /*
33 Enter the number of elements in the array: 5
34 Enter 5 elements:
35 1 2 3 4 5
36 Sum of all elements = 15
37 */
```



```
1 //
2 // 2. Write a C program to swap value of two variables using pointer.
3 //
4 #include <stdio.h>
5
6 // Function to swap two values using pointers
7 void swap(int *a, int *b) {
8     int temp;
9     temp = *a;
10    *a = *b;
11    *b = temp;
12 }
13
14 int main() {
15     int x, y;
16
17     // Input values
18     printf("Enter value of x: ");
19     scanf("%d", &x);
20
21     printf("Enter value of y: ");
22     scanf("%d", &y);
23
24     // Before swapping
25     printf("Before swapping: x = %d, y = %d\n", x, y);
26
27     // Call swap function
28     swap(&x, &y);
29
30     // After swapping
31     printf("After swapping: x = %d, y = %d\n", x, y);
32
33     return 0;
34 }
35 /*
36 Enter value of x: 10
37 Enter value of y: 20
38 Before swapping: x = 10, y = 20
39 After swapping: x = 20, y = 10
40 */
```

```
1 //
2 // 3. Write a C program to add two numbers using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int a, b, sum;
8     int *ptr1, *ptr2;
9
10    // Input numbers
11    printf("Enter first number: ");
12    scanf("%d", &a);
13
14    printf("Enter second number: ");
15    scanf("%d", &b);
16
17    // Assign addresses to pointers
18    ptr1 = &a;
19    ptr2 = &b;
20
21    // Add values using pointers
22    sum = *ptr1 + *ptr2;
23
24    // Output the result
25    printf("Sum = %d\n", sum);
26
27    return 0;
28 }
29 /*
30 Enter first number: 12
31 Enter second number: 8
32 Sum = 20
33 */
```

```

1 //
2 //4. Write a C program to input and print array elements using pointer.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n;
8     int *ptr;
9
10    // Input the number of elements
11    printf("Enter number of elements in the array: ");
12    scanf("%d", &n);
13
14    // Pointer points to the array
15    ptr = arr;
16
17    // Input array elements using pointer
18    printf("Enter %d elements:\n", n);
19    for (int i = 0; i < n; i++) {
20        scanf("%d", (ptr + i)); // same as &arr[i]
21    }
22
23    // Print array elements using pointer
24    printf("Array elements are:\n");
25    for (int i = 0; i < n; i++) {
26        printf("%d ", *(ptr + i)); // same as arr[i]
27    }
28
29    printf("\n");
30
31    return 0;
32 }
33 /*
34 Enter number of elements in the array: 5
35 Enter 5 elements:
36 10 20 30 40 50
37 Array elements are:
38 10 20 30 40 50
39 */

```

```

1 //
2 //5. Write a C program to copy one array to another using pointer.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int source[100], destination[100], n;
8     int *srcPtr, *destPtr;
9
10    // Input array size
11    printf("Enter the number of elements: ");
12    scanf("%d", &n);
13
14    // Input elements in source array
15    printf("Enter %d elements for source array:\n", n);
16    for (int i = 0; i < n; i++) {
17        scanf("%d", &source[i]);
18    }
19
20    // Initialize pointers
21    srcPtr = source; // pointer to source array
22    destPtr = destination; // pointer to destination array
23
24    // Copy elements using pointers
25    for (int i = 0; i < n; i++) {
26        *(destPtr + i) = *(srcPtr + i);
27    }
28
29    // Print destination array
30    printf("Elements in destination array:\n");
31    for (int i = 0; i < n; i++) {
32        printf("%d ", destination[i]);
33    }
34
35    printf("\n");
36
37    return 0;
38 }
39 /*
40 Enter the number of elements: 5
41 Enter 5 elements for source array:
42 1 2 3 4 5
43 Elements in destination array:
44 1 2 3 4 5
45 */

```

```

1 //
2 //6. Write a C program to swap two arrays using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr1[100], arr2[100], temp, n;
8     int *ptr1, *ptr2;
9
10    // Input array size
11    printf("Enter the number of elements in the arrays: ");
12    scanf("%d", &n);
13
14    // Input elements for arr1
15    printf("Enter %d elements for first array:\n", n);
16    for (int i = 0; i < n; i++) {
17        scanf("%d", &arr1[i]);
18    }
19
20    // Input elements for arr2
21    printf("Enter %d elements for second array:\n", n);
22    for (int i = 0; i < n; i++) {
23        scanf("%d", &arr2[i]);
24    }
25
26    // Set pointers
27    ptr1 = arr1;
28    ptr2 = arr2;
29
30    // Swap elements using pointers
31    for (int i = 0; i < n; i++) {
32        temp = *(ptr1 + i);
33        *(ptr1 + i) = *(ptr2 + i);
34        *(ptr2 + i) = temp;
35    }
36
37    // Print swapped arrays
38    printf("After swapping:\n");
39    printf("First array: ");
40    for (int i = 0; i < n; i++) {
41        printf("%d ", arr1[i]);
42    }
43
44    printf("\nSecond array: ");
45    for (int i = 0; i < n; i++) {
46        printf("%d ", arr2[i]);
47    }
48
49    printf("\n");
50
51    return 0;
52 }
53 /*
54 Enter the number of elements in the arrays: 3
55 Enter 3 elements for first array:
56 1 2 3
57 Enter 3 elements for second array:

```

58 4 5 6

59 *After swapping:*

60 *First array: 4 5 6*

61 *Second array: 1 2 3*

62 \*/

```

1 //
2 //7. Write a C program to reverse an array using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, temp;
8     int *start, *end;
9
10    // Input number of elements
11    printf("Enter the number of elements: ");
12    scanf("%d", &n);
13
14    // Input array elements
15    printf("Enter %d elements:\n", n);
16    for (int i = 0; i < n; i++) {
17        scanf("%d", &arr[i]);
18    }
19
20    // Set start and end pointers
21    start = arr;           // pointing to the first element
22    end = arr + n - 1;    // pointing to the last element
23
24    // Reverse using pointers
25    while (start < end) {
26        temp = *start;
27        *start = *end;
28        *end = temp;
29
30        start++;
31        end--;
32    }
33
34    // Print reversed array
35    printf("Reversed array:\n");
36    for (int i = 0; i < n; i++) {
37        printf("%d ", arr[i]);
38    }
39
40    printf("\n");
41
42    return 0;
43 }
44 /*
45 Enter the number of elements: 5
46 Enter 5 elements:
47 10 20 30 40 50
48 Reversed array:
49 50 40 30 20 10
50 */

```

```

1 //
2 //8. Write a C program to search for an element in array using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int arr[100], n, key, found = 0;
8     int *ptr;
9
10    // Input array size
11    printf("Enter the number of elements: ");
12    scanf("%d", &n);
13
14    // Input array elements
15    printf("Enter %d elements:\n", n);
16    for (int i = 0; i < n; i++) {
17        scanf("%d", &arr[i]);
18    }
19
20    // Input element to search
21    printf("Enter the element to search: ");
22    scanf("%d", &key);
23
24    // Pointer to the start of the array
25    ptr = arr;
26
27    // Search using pointer
28    for (int i = 0; i < n; i++) {
29        if (*(ptr + i) == key) {
30            printf("Element %d found at position %d (index %d).\n", key, i + 1, i);
31            found = 1;
32            break;
33        }
34    }
35
36    if (!found) {
37        printf("Element %d not found in the array.\n", key);
38    }
39
40    return 0;
41 }
42 /*
43 Enter the number of elements: 5
44 Enter 5 elements:
45 10 20 30 40 50
46 Enter the element to search: 30
47 Element 30 found at position 3 (index 2).
48 */

```



```

1 //
2 //9. Write a C program to add two 2 X 2 matrix using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int mat1[2][2], mat2[2][2], sum[2][2];
8     int *ptr1, *ptr2, *ptrSum;
9
10    // Input elements for Matrix 1
11    printf("Enter elements of first 2x2 matrix:\n");
12    for (int i = 0; i < 2; i++) {
13        for (int j = 0; j < 2; j++) {
14            scanf("%d", &mat1[i][j]);
15        }
16    }
17
18    // Input elements for Matrix 2
19    printf("Enter elements of second 2x2 matrix:\n");
20    for (int i = 0; i < 2; i++) {
21        for (int j = 0; j < 2; j++) {
22            scanf("%d", &mat2[i][j]);
23        }
24    }
25
26    // Use pointers to add matrices
27    ptr1 = &mat1[0][0];
28    ptr2 = &mat2[0][0];
29    ptrSum = &sum[0][0];
30
31    for (int i = 0; i < 2; i++) {
32        *(ptrSum + i) = *(ptr1 + i) + *(ptr2 + i);
33    }
34
35    // Print the resulting sum matrix
36    printf("Sum of the two matrices:\n");
37    for (int i = 0; i < 2; i++) {
38        for (int j = 0; j < 2; j++) {
39            printf("%d ", sum[i][j]);
40        }
41        printf("\n");
42    }
43
44    return 0;
45 }
46
47 /*
48 Enter elements of first 2x2 matrix:
49 1 2
50 3 4
51 Enter elements of second 2x2 matrix:
52 5 6
53 7 8
54 Sum of the two matrices:
55 6 8
56 10 12
57 */

```

```

1 //
2 //10. Write a C program to multiply two 2 X 2 matrix using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     int mat1[2][2], mat2[2][2], result[2][2];
8     int *a, *b, *res;
9
10    // Input first matrix
11    printf("Enter elements of first 2x2 matrix:\n");
12    for (int i = 0; i < 2; i++)
13        for (int j = 0; j < 2; j++)
14            scanf("%d", &mat1[i][j]);
15
16    // Input second matrix
17    printf("Enter elements of second 2x2 matrix:\n");
18    for (int i = 0; i < 2; i++)
19        for (int j = 0; j < 2; j++)
20            scanf("%d", &mat2[i][j]);
21
22    // Pointers to matrices
23    a = &mat1[0][0];
24    b = &mat2[0][0];
25    res = &result[0][0];
26
27    // Matrix multiplication using pointers
28    for (int i = 0; i < 2; i++) {
29        for (int j = 0; j < 2; j++) {
30            *(res + i * 2 + j) = 0;
31            for (int k = 0; k < 2; k++) {
32                *(res + i * 2 + j) += (*(a + i * 2 + k)) * (*(b + k * 2 + j));
33            }
34        }
35    }
36
37    // Print result matrix
38    printf("Resultant matrix after multiplication:\n");
39    for (int i = 0; i < 2; i++) {
40        for (int j = 0; j < 2; j++) {
41            printf("%d ", result[i][j]);
42        }
43        printf("\n");
44    }
45
46    return 0;
47 }
48 /*
49 Enter elements of first 2x2 matrix:
50 1 2
51 3 4
52 Enter elements of second 2x2 matrix:
53 5 6
54 7 8
55 Resultant matrix after multiplication:
56 19 22
57 43 50

```



```
1 //
2 //11. Write a C program to find length of string using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char str[100];
8     char *ptr;
9     int length = 0;
10
11     // Input string
12     printf("Enter a string: ");
13     fgets(str, sizeof(str), stdin); // safer than gets()
14
15     // Initialize pointer to the beginning of the string
16     ptr = str;
17
18     // Traverse the string using pointer until null character
19     while (*ptr != '\0' && *ptr != '\n') {
20         length++;
21         ptr++;
22     }
23
24     printf("Length of the string = %d\n", length);
25
26     return 0;
27 }
28 /*
29 Enter a string: Hello
30 Length of the string = 5
31 */
```

```
1 //
2 //12. Write a C program to copy one string to another using pointer.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char source[100], destination[100];
8     char *srcPtr, *destPtr;
9
10    // Input the source string
11    printf("Enter the source string: ");
12    fgets(source, sizeof(source), stdin); // safer than gets()
13
14    // Initialize pointers
15    srcPtr = source;
16    destPtr = destination;
17
18    // Copy the string using pointers
19    while (*srcPtr != '\0' && *srcPtr != '\n') {
20        *destPtr = *srcPtr;
21        srcPtr++;
22        destPtr++;
23    }
24
25    // Null-terminate the destination string
26    *destPtr = '\0';
27
28    // Print the copied string
29    printf("Destination string: %s\n", destination);
30
31    return 0;
32 }
33 /*
34 Enter the source string: Hello, world!
35 Destination string: Hello, world!
36 */
```

```

1 //
2 //13. Write a C program to concatenate two strings using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char str1[100], str2[100];
8     char *ptr1, *ptr2;
9
10    // Input the two strings
11    printf("Enter the first string: ");
12    fgets(str1, sizeof(str1), stdin); // safer than gets()
13
14    printf("Enter the second string: ");
15    fgets(str2, sizeof(str2), stdin); // safer than gets()
16
17    // Initialize pointers to the first string and second string
18    ptr1 = str1;
19    ptr2 = str2;
20
21    // Move ptr1 to the end of the first string
22    while (*ptr1 != '\0') {
23        ptr1++;
24    }
25
26    // Concatenate the second string to the first string
27    while (*ptr2 != '\0' && *ptr2 != '\n') {
28        *ptr1 = *ptr2;
29        ptr1++;
30        ptr2++;
31    }
32
33    // Null-terminate the concatenated string
34    *ptr1 = '\0';
35
36    // Print the concatenated string
37    printf("Concatenated string: %s\n", str1);
38
39    return 0;
40 }
41 /*
42 Enter the first string: Hello
43 Enter the second string: World
44 Concatenated string: HelloWorld
45 */

```

```

1 //
2 //14. Write a C program to compare two strings using pointers.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char str1[100], str2[100];
8     char *ptr1, *ptr2;
9
10    // Input the two strings
11    printf("Enter the first string: ");
12    fgets(str1, sizeof(str1), stdin); // safer than gets()
13
14    printf("Enter the second string: ");
15    fgets(str2, sizeof(str2), stdin); // safer than gets()
16
17    // Initialize pointers
18    ptr1 = str1;
19    ptr2 = str2;
20
21    // Compare the two strings using pointers
22    while (*ptr1 != '\0' && *ptr2 != '\0') {
23        if (*ptr1 != *ptr2) {
24            printf("Strings are not equal.\n");
25            return 0;
26        }
27        ptr1++;
28        ptr2++;
29    }
30
31    // Check if both strings have reached the null character
32    if (*ptr1 == '\0' && *ptr2 == '\0') {
33        printf("Strings are equal.\n");
34    } else {
35        printf("Strings are not equal.\n");
36    }
37
38    return 0;
39 }
40 /*
41 Enter the first string: Hello
42 Enter the second string: Hello
43 Strings are equal.
44
45 Enter the first string: Hello
46 Enter the second string: World
47 Strings are not equal.
48 */

```

```

1 //
2 //15. Write a C program to find a substring from a given string using pointers.
3 //
4 #include <stdio.h>
5
6 int findSubstring(char *str, char *substr) {
7     char *ptr1, *ptr2;
8     int index = -1;
9
10    // Traverse through the main string
11    for (ptr1 = str; *ptr1 != '\0'; ptr1++) {
12        // Check if the substring matches starting from this position
13        ptr2 = substr;
14        if (*ptr1 == *ptr2) {
15            char *temp1 = ptr1, *temp2 = ptr2;
16
17            // Compare the rest of the characters
18            while (*temp1 != '\0' && *temp2 != '\0' && *temp1 == *temp2) {
19                temp1++;
20                temp2++;
21            }
22
23            // If end of substring is reached, match found
24            if (*temp2 == '\0') {
25                index = ptr1 - str;
26                return index; // Return the starting index of the substring
27            }
28        }
29    }
30
31    return index; // Return -1 if substring not found
32 }
33
34 int main() {
35     char str[100], substr[50];
36     int position;
37
38     // Input the main string and substring
39     printf("Enter the main string: ");
40     fgets(str, sizeof(str), stdin); // safer than gets()
41
42     printf("Enter the substring to find: ");
43     fgets(substr, sizeof(substr), stdin); // safer than gets()
44
45     // Remove newline character from fgets input
46     str[strcspn(str, "\n")] = '\0';
47     substr[strcspn(substr, "\n")] = '\0';
48
49     // Call the function to find the substring
50     position = findSubstring(str, substr);
51
52     // Output result
53     if (position != -1) {
54         printf("Substring found at position %d.\n", position);
55     } else {
56         printf("Substring not found.\n");
57     }

```



```
58 //
59     return 0;
60 }
61
62 /*
63  Enter the main string: This is a sample string.
64  Enter the substring to find: sample
65  Substring found at position 10.
66
67  Enter the main string: This is a sample string.
68  Enter the substring to find: hello
69  Substring not found.
70  */
```

# Assignment – 5

## Topic: File Handling

1. Write a C Program to list all files and sub-directories in a directory.
2. Write a C Program to count number of lines in a file.
3. Write a C Program to print contents of file.
4. Write a C Program to copy contents of one file to another file.
5. Write a C Program to merge contents of two files into a third file.
6. Write a C program to delete a file.

```

1 //
2 //1. Write a C Program to list all files and sub-directories in a directory.
3 //
4 #include <stdio.h>
5 #include <dirent.h>
6 #include <stdlib.h>
7
8 void listFilesAndDirectories(const char *dirPath) {
9     struct dirent *entry;
10    DIR *dir = opendir(dirPath);
11
12    // Check if directory can be opened
13    if (dir == NULL) {
14        perror("opendir");
15        return;
16    }
17
18    printf("Listing files and subdirectories in '%s':\n", dirPath);
19
20    // Read and print directory entries
21    while ((entry = readdir(dir)) != NULL) {
22        // Skip the "." and ".." directories
23        if (entry->d_name[0] != '.') {
24            printf("%s\n", entry->d_name);
25        }
26    }
27
28    // Close the directory
29    closedir(dir);
30 }
31
32 int main() {
33     char dirPath[256];
34
35     // Input the directory path
36     printf("Enter directory path: ");
37     fgets(dirPath, sizeof(dirPath), stdin);
38
39     // Remove trailing newline character from fgets input
40     dirPath[strcspn(dirPath, "\n")] = '\0';
41
42     // Call function to list files and subdirectories
43     listFilesAndDirectories(dirPath);
44
45     return 0;
46 }
47 /*
48 Enter directory path:D:\OneDrive - uem.edu.in\ODD SEMESTER - 2025-26\Class Slides and
Assignments\MCA192-C Programming Lab\Assignment - 5
49
50 Listing files and subdirectories in 'D:\OneDrive - uem.edu.in\ODD SEMESTER - 2025-26\
Class Slides and Assignments\MCA192
51 -C Programming Lab\Assignment - 5':
52 A5Q1-Directory_Subdir.c
53 A5Q1-Directory_Subdir.exe
54 */

```

```

1 //
2 //2. Write a C Program to count number of lines in a file.
3 //
4 #include <stdio.h>
5
6 int main() {
7     FILE *file;
8     char filename[100];
9     char ch;
10    int lineCount = 0;
11
12    // Ask user for file name
13    printf("Enter the file name: ");
14    fgets(filename, sizeof(filename), stdin);
15    filename[strcspn(filename, "\n")] = '\0'; // Remove newline from filename
16
17    // Open file in read mode
18    file = fopen(filename, "r");
19
20    // Check if file opened successfully
21    if (file == NULL) {
22        perror("Error opening file");
23        return 1;
24    }
25
26    // Read character by character and count newlines
27    while ((ch = fgetc(file)) != EOF) {
28        if (ch == '\n') {
29            lineCount++;
30        }
31    }
32
33    // If the file isn't empty, count at least one line
34    if (lineCount == 0 && !feof(file)) {
35        lineCount = 1;
36    }
37
38    // Close the file
39    fclose(file);
40
41    // Print the result
42    printf("Number of lines in the file: %d\n", lineCount);
43
44    return 0;
45 }
46 /*
47 Enter the file name:A5Q1-Directory_Subdir.c
48
49 Number of lines in the file: 53
50 */

```

```

1 //
2 //3. Write a C Program to print contents of file.
3 //
4 #include <stdio.h>
5
6 int main() {
7     FILE *file;
8     char filename[100];
9     char ch;
10
11     // Ask user for the file name
12     printf("Enter the file name: ");
13     fgets(filename, sizeof(filename), stdin);
14     filename[strcspn(filename, "\n")] = '\0'; // Remove newline from input
15
16     // Open the file in read mode
17     file = fopen(filename, "r");
18
19     // Check if file exists
20     if (file == NULL) {
21         perror("Error opening file");
22         return 1;
23     }
24
25     // Print the contents of the file character by character
26     printf("\nContents of the file '%s':\n\n", filename);
27     while ((ch = fgetc(file)) != EOF) {
28         putchar(ch);
29     }
30
31     // Close the file
32     fclose(file);
33
34     return 0;
35 }
36 /*
37 Enter the file name:Assignment-5.txt
38
39 Contents of the file 'Assignment-5.txt':
40
41 Assignment: 5
42 Topic: File Handling
43 1. Write a C Program to list all files and sub-directories in a directory.
44 2. Write a C Program to count number of lines in a file.
45 3. Write a C Program to print contents of file.
46 4. Write a C Program to copy contents of one file to another file.
47 5. Write a C Program to merge contents of two files into a third file.
48 6. Write a C program to delete a file.
49 */

```

```
1 //
2 // Created by ankur on 14-04-2025.
3 //
4 #include <stdio.h>
5
6 int main() {
7     FILE *sourceFile, *destFile;
8     char sourceName[100], destName[100];
9     char ch;
10
11     // Get source file name
12     printf("Enter the source file name: ");
13     fgets(sourceName, sizeof(sourceName), stdin);
14     sourceName[strcspn(sourceName, "\n")] = '\0'; // Remove newline
15
16     // Get destination file name
17     printf("Enter the destination file name: ");
18     fgets(destName, sizeof(destName), stdin);
19     destName[strcspn(destName, "\n")] = '\0'; // Remove newline
20
21     // Open source file in read mode
22     sourceFile = fopen(sourceName, "r");
23     if (sourceFile == NULL) {
24         perror("Error opening source file");
25         return 1;
26     }
27
28     // Open destination file in write mode
29     destFile = fopen(destName, "w");
30     if (destFile == NULL) {
31         perror("Error opening destination file");
32         fclose(sourceFile);
33         return 1;
34     }
35
36     // Copy contents character by character
37     while ((ch = fgetc(sourceFile)) != EOF) {
38         fputc(ch, destFile);
39     }
40
41     printf("File copied successfully from '%s' to '%s'.\n", sourceName, destName);
42
43     // Close both files
44     fclose(sourceFile);
45     fclose(destFile);
46
47     return 0;
48 }
49 /*
50 Enter the source file name:Assignment-5.txt
51
52 Enter the destination file name:Hello.txt
53
54 File copied successfully from 'Assignment-5.txt' to 'Hello.txt'.
55 */
```

```

1 //
2 //5. Write a C Program to merge contents of two files into a third file.
3 //
4 #include <stdio.h>
5
6 void copyContents(FILE *source, FILE *dest) {
7     char ch;
8     while ((ch = fgetc(source)) != EOF) {
9         fputc(ch, dest);
10    }
11 }
12
13 int main() {
14     FILE *file1, *file2, *file3;
15     char filename1[100], filename2[100], filename3[100];
16
17     // Get the filenames
18     printf("Enter first source file name: ");
19     fgets(filename1, sizeof(filename1), stdin);
20     filename1[strcspn(filename1, "\n")] = '\0';
21
22     printf("Enter second source file name: ");
23     fgets(filename2, sizeof(filename2), stdin);
24     filename2[strcspn(filename2, "\n")] = '\0';
25
26     printf("Enter destination file name: ");
27     fgets(filename3, sizeof(filename3), stdin);
28     filename3[strcspn(filename3, "\n")] = '\0';
29
30     // Open the files
31     file1 = fopen(filename1, "r");
32     file2 = fopen(filename2, "r");
33     file3 = fopen(filename3, "w");
34
35     // Check for file open errors
36     if (file1 == NULL || file2 == NULL || file3 == NULL) {
37         perror("Error opening files");
38         return 1;
39     }
40
41     // Copy contents of first file
42     copyContents(file1, file3);
43
44     // Copy contents of second file
45     copyContents(file2, file3);
46
47     printf("Files '%s' and '%s' merged into '%s' successfully.\n", filename1, filename2,
filename3);
48
49     // Close all files
50     fclose(file1);
51     fclose(file2);
52     fclose(file3);
53
54     return 0;
55 }
56

```

57 *///  
\**

58 *Enter first source file name:A5Q5-Merge\_File.c*

59

60 *Enter second source file name:Assignment-5.txt*

61

62 *Enter destination file name:Merge\_Files.txt*

63

64 *Files 'A5Q5-Merge\_File.c' and 'Assignment-5.txt' merged into 'Merge\_Files.txt' successfully.*

65 *\*/*

66



```
1 //
2 //6. Write a C program to delete a file.
3 //
4 #include <stdio.h>
5
6 int main() {
7     char filename[100];
8
9     // Ask user for the filename to delete
10    printf("Enter the file name to delete: ");
11    fgets(filename, sizeof(filename), stdin);
12    filename[strcspn(filename, "\n")] = '\0'; // Remove newline character
13
14    // Try to delete the file
15    if (remove(filename) == 0) {
16        printf("File '%s' deleted successfully.\n", filename);
17    } else {
18        perror("Error deleting file");
19    }
20
21    return 0;
22 }
23
24 /*
25 Enter the file name to delete:Assignment-5_Copy.txt
26
27 File 'Assignment-5_Copy.txt' deleted successfully.
28 */
```