ASSIGNMENT – 1

ROLL NO: 27

NAME: SHINGALA NISHIT

CLASS: MCA 1

SUBJECT: DATA STRUCTURES

1. **WAP to use binary operator + add two object of class Numbers having num1 and num2 as its data members and display result.**

```
// LINEAR
#include <stdio.h>

int linearSearch(int arr[], int n, int key, int *comparisons) {
    for (int i = 0; i < n; i++) {
        (*comparisons)++;           // count each comparison
        if (arr[i] == key) {
            return i;               // found at index i
        }
    }
    return -1;                      // not found
}

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 30;
    int comparisons = 0;

    int result = linearSearch(arr, n, key, &comparisons);

    if (result != -1)
        printf("Linear Search: %d found at index %d\n", key, result);
    else
        printf("Linear Search: %d not found\n", key);

    printf("Number of comparisons (Linear Search): %d\n", comparisons);

    return 0;
}
```

```
// BINARY
#include <stdio.h>
int binarySearch(int arr[], int n, int key, int *comparisons) {
    int low = 0, high = n - 1;

    while (low <= high) {
        int mid = (low + high) / 2;
        (*comparisons)++;               // count each comparison

        if (arr[mid] == key)
            return mid;                 // found
        else if (arr[mid] < key)
            low = mid + 1;              // search in right half
        else
            high = mid - 1;             // search in left half
    }
    return -1;                          // not found
}

int main() {
    int arr[] = {10, 20, 30, 40, 50};  // sorted array
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 30;
    int comparisons = 0;

    int result = binarySearch(arr, n, key, &comparisons);

    if (result != -1)
        printf("Binary Search: %d found at index %d\n", key, result);
    else
        printf("Binary Search: %d not found\n", key);

    printf("Number of comparisons (Binary Search): %d\n", comparisons);

    return 0;
}
```

OUTPUT:

```
Linear Search: 30 found at index 2
Number of comparisons (Linear Search): 3

Binary Search: 30 found at index 2
Number of comparisons (Binary Search): 1
```

2. Write a program to find the memory address of any element in the array.

```
#include <stdio.h>

int main() {
    int n, i, pos;

    printf("Enter number of elements in array: ");
    scanf("%d", &n);

    int arr[n];  // array declaration

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the position (index) of element to find its address: ");
    scanf("%d", &pos);

    if (pos >= 0 && pos < n) {
        printf("Element at index %d = %d\n", pos, arr[pos]);
        printf("Address of arr[%d] = %p\n", pos, (void*)&arr[pos]);
    } else {
        printf("Invalid index! Valid range is 0 to %d\n", n - 1);
    }

    return 0;
}
```

OUTPUT:

```
Enter number of elements in array: 5
Enter 5 elements:
10 20 30 40 50
Enter the position (index) of element to find its address: 3
Element at index 3 = 40
Address of arr[3] = 0x7ffcdbdbf0ac
```

3. Write a program to perform insertion and deletion in the array at start of array, end of array and middle of array.

```
#include <stdio.h>

#define MAX 100   // maximum array size

// Function to display the array
void display(int arr[], int n) {
   if (n == 0) {
      printf("Array is empty!\n");
      return;
   }
   printf("Array elements: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
}

int main() {
   int arr[MAX], n, choice, pos, value;

   printf("Enter number of elements in array: ");
   scanf("%d", &n);

   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   while (1) {
      printf("\n--- Menu ---\n");
      printf("1. Insert at Start\n");
      printf("2. Insert at End\n");
      printf("3. Insert at Middle (position)\n");
      printf("4. Delete from Start\n");
      printf("5. Delete from End\n");
      printf("6. Delete from Middle (position)\n");
      printf("7. Display Array\n");
printf("8. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
    case 1: // Insert at Start
        printf("Enter value to insert: ");
        scanf("%d", &value);
        for (int i = n; i > 0; i--) {
            arr[i] = arr[i - 1];
        }
        arr[0] = value;
        n++;
        break;

    case 2: // Insert at End
        printf("Enter value to insert: ");
        scanf("%d", &value);
        arr[n] = value;
        n++;
        break;

    case 3: // Insert at Middle (given position)
        printf("Enter position (0 to %d): ", n);
        scanf("%d", &pos);
        if (pos < 0 || pos > n) {
            printf("Invalid position!\n");
        } else {
            printf("Enter value to insert: ");
            scanf("%d", &value);
            for (int i = n; i > pos; i--) {
                arr[i] = arr[i - 1];
            }
            arr[pos] = value;
            n++;
        }
        break;

    case 4: // Delete from Start
        if (n == 0) {
            printf("Array is empty!\n");
```

```
                } else {
                    for (int i = 0; i < n - 1; i++) {
                        arr[i] = arr[i + 1];
                    }
                    n--;
                }
                break;

            case 5: // Delete from End
                if (n == 0) {
                    printf("Array is empty!\n");
                } else {
                    n--;
                }
                break;

            case 6: // Delete from Middle
                printf("Enter position (0 to %d): ", n - 1);
                scanf("%d", &pos);
                if (pos < 0 || pos >= n) {
                    printf("Invalid position!\n");
                } else {
                    for (int i = pos; i < n - 1; i++) {
                        arr[i] = arr[i + 1];
                    }
                    n--;
                }
                break;

            case 7:
                display(arr, n);
                break;

            case 8:
                return 0;

            default:
                printf("Invalid choice!\n");
        }
    }
}
```

OUTPUT:

```
Enter number of elements in array: 5
Enter 5 elements:
10 20 30 40 50

--- Menu ---
1. Insert at Start
2. Insert at End
3. Insert at Middle (position)
4. Delete from Start
5. Delete from End
6. Delete from Middle (position)
7. Display Array
8. Exit
Enter your choice: 1
Enter value to insert: 5
Array elements: 5 10 20 30 40 50
```

4. Write a program to merge two arrays when they are sorted and when they are not sorted.

```
#include <stdio.h>

// Function to display array
void display(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// Function to merge two sorted arrays
void mergeSorted(int arr1[], int n1, int arr2[], int n2, int merged[]) {
    int i = 0, j = 0, k = 0;

    while (i < n1  && j < n2) {
        if (arr1[i] < arr2[j])
            merged[k++] = arr1[i++];
        else
            merged[k++] = arr2[j++];
    }

    while (i < n1)  // remaining elements of arr1
        merged[k++] = arr1[i++];

    while (j < n2)  // remaining elements of arr2
        merged[k++] = arr2[j++];
}

// Function to merge two unsorted arrays
void mergeUnsorted(int arr1[], int n1, int arr2[], int n2, int merged[]) {
    int i, j;
    // Copy arr1
    for (i = 0; i < n1; i++)
        merged[i] = arr1[i];
    // Copy arr2
    for (j = 0; j < n2; j++)
        merged[i + j] = arr2[j];
```

```
}

// Simple bubble sort for unsorted merge
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
int main() {
    int arr1[50], arr2[50], merged[100];
    int n1, n2, choice;

    printf("Enter size of first array: ");
    scanf("%d", &n1);
    printf("Enter elements of first array:\n");
    for (int i = 0; i < n1; i++)
        scanf("%d", &arr1[i]);

    printf("Enter size of second array: ");
    scanf("%d", &n2);
    printf("Enter elements of second array:\n");
    for (int i = 0; i < n2; i++)
        scanf("%d", &arr2[i]);

    printf("\nChoose option:\n");
    printf("1. Merge when arrays are sorted\n");
    printf("2. Merge when arrays are unsorted\n");
    scanf("%d", &choice);

    if (choice == 1) {
        mergeSorted(arr1, n1, arr2, n2, merged);
        printf("Merged Sorted Array:\n");
        display(merged, n1 + n2);
    }
    else if (choice == 2) {
        mergeUnsorted(arr1, n1, arr2, n2, merged);
        bubbleSort(merged, n1 + n2);
        printf("Merged & Sorted Array (unsorted input):\n");
        display(merged, n1 + n2);
    }
    else {
        printf("Invalid choice!\n");
    }

    return 0;
}
```

OUTPUT:

```
Enter size of first array: 4
Enter elements: 1 3 5 7
Enter size of second array: 3
Enter elements: 2 4 6
Choice: 1
Merged Sorted Array:
1 2 3 4 5 6 7

Enter size of first array: 3
Enter elements: 7 1 5
Enter size of second array: 4
Enter elements: 2 8 3 6
Choice: 2
Merged & Sorted Array (unsorted input):
1 2 3 5 6 7 8
```

5. Consider an array MARKS [20] [5] which stores the marks obtained by 20 students in 5 subjects. Now write a program to
(a) find the average marks obtained in each subject..
(b) find the average marks obtained by every student.
(c) find the number of students who have scored below 50 in their average.
(d) display the scores obtained by every student.

```
#include <stdio.h>

#define STUDENTS 20
#define SUBJECTS 5

int main() {
    int MARKS[STUDENTS][SUBJECTS];
    float avgSubject[SUBJECTS] = {0};   // store average of each subject
    float avgStudent[STUDENTS] = {0};   // store average of each student
    int below50 = 0;

    // Input marks
    printf("Enter marks of %d students in %d subjects:\n", STUDENTS, SUBJECTS);
    for (int i = 0; i < STUDENTS; i++) {
        printf("Student %d:\n", i + 1);
        for (int j = 0; j < SUBJECTS; j++) {
            printf("  Subject %d: ", j + 1);
            scanf("%d", &MARKS[i][j]);
        }
    }

    // (a) Average of each subject
    for (int j = 0; j < SUBJECTS; j++) {
        int sum = 0;
        for (int i = 0; i < STUDENTS; i++) {
            sum += MARKS[i][j];
        }
        avgSubject[j] = (float)sum / STUDENTS;
    }

    // (b) Average of each student
    for (int i = 0; i < STUDENTS; i++) {
        int sum = 0;
        for (int j = 0; j < SUBJECTS; j++) {
            sum += MARKS[i][j];
        }
        avgStudent[i] = (float)sum / SUBJECTS;

        // (c) Count students below 50 average
        if (avgStudent[i] < 50)
            below50++;
    }

    // (d) Display scores of each student
    printf("\n--- Scores of Each Student ---\n");
    for (int i = 0; i < STUDENTS; i++) {
        printf("Student %d: ", i + 1);
        for (int j = 0; j < SUBJECTS; j++) {
            printf("%d ", MARKS[i][j]);
        }
        printf("\n");
    }

    // Display average per subject
    printf("\n--- Average Marks of Each Subject ---\n");
    for (int j = 0; j < SUBJECTS; j++) {
        printf("Subject %d: %.2f\n", j + 1, avgSubject[j]);
    }

    // Display average per student
    printf("\n--- Average Marks of Each Student ---\n");
    for (int i = 0; i < STUDENTS; i++) {
        printf("Student %d: %.2f\n", i + 1, avgStudent[i]);
    }

    // Display number of students below 50 average
    printf("\nNumber of students with average marks below 50 = %d\n", below50);

    return 0;
}
```

OUTPUT:

```
Enter marks of 20 students in 5 subjects:
Student 1:
  Subject 1: 40
  Subject 2: 50
  Subject 3: 60
  Subject 4: 70
  Subject 5: 80
...

--- Scores of Each Student ---
Student 1: 40 50 60 70 80
...

--- Average Marks of Each Subject ---
Subject 1: 52.30
Subject 2: 60.25
...

--- Average Marks of Each Student ---
Student 1: 60.00
...

Number of students with average marks below 50 = 7
```

6. **Write a program that reads an array of 100 integers. Display all the pairs of elements whose sum is 50.**

```c
#include <stdio.h>

#define SIZE 100

int main() {
    int arr[SIZE];
    int i, j;

    // Input array
    printf("Enter %d integers:\n", SIZE);
    for (i = 0; i < SIZE; i++) {
        scanf("%d", &arr[i]);
    }

    printf("\nPairs with sum 50:\n");
    int found = 0;
    for (i = 0; i < SIZE; i++) {
        for (j = i + 1; j < SIZE; j++) {   // avoid duplicate pairs
            if (arr[i] + arr[j] == 50) {
                printf("(%d, %d)\n", arr[i], arr[j]);
                found = 1;
            }
        }
    }

    if (!found)
        printf("No pairs found!\n");

    return 0;
}
```

OUTPUT:

```
10 20 30 40 15 35 5 45 ... (rest numbers)
Pairs with sum 50:
(10, 40)
(20, 30)
(15, 35)
(5, 45)
```

7. Write a program to interchange the second element with the second last element.

```c
#include <stdio.h>

int main() {
    int arr[100], n, temp;

    // Input array size
    printf("Enter number of elements: ");
    scanf("%d", &n);

    // Input elements
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Interchange second and second last elements
    if (n >= 2) {
        temp = arr[1];
        arr[1] = arr[n - 2];
        arr[n - 2] = temp;
    }

    // Display updated array
    printf("\nArray after interchange:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

```
Enter number of elements: 6
Enter 6 elements:
10 20 30 40 50 60

Array after interchange:
10 50 30 40 20 60
```

8. Write a program that calculates the sum of squares of the elements.

```
#include <stdio.h>

int main() {
    int arr[100], n;
    long long sum = 0; // use long long for large values

    // Input array size
    printf("Enter number of elements: ");
    scanf("%d", &n);

    // Input elements
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Calculate sum of squares
    for (int i = 0; i < n; i++) {
        sum += (long long)arr[i] * arr[i];
    }

    // Display result
    printf("\nSum of squares of elements = %lld\n", sum);

    return 0;
}
```

OUTPUT:

```
Enter number of elements: 5
Enter 5 elements:
1 2 3 4 5

Sum of squares of elements = 55
```

9. Write a program to compute the sum and mean of the elements of a two-dimensional array.

```
#include <stdio.h>

int main() {
    int rows, cols;
    int arr[50][50];   // max 50x50
    int sum = 0;
    float mean;

    // Input rows and columns
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    printf("Enter number of columns: ");
    scanf("%d", &cols);

    // Input elements
    printf("Enter elements of the array:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &arr[i][j]);
            sum += arr[i][j];  // accumulate sum
        }
    }

    // Calculate mean
    mean = (float)sum / (rows * cols);

    // Output
    printf("\nSum of all elements = %d\n", sum);
    printf("Mean of all elements = %.2f\n", mean);

    return 0;
}
```

OUTPUT:

```
Enter number of rows: 2
Enter number of columns: 3
Enter elements of the array:
1 2 3
4 5 6

Sum of all elements = 21
Mean of all elements = 3.50
```

10. Write a program to read and display a square (using functions).

```
#include <stdio.h>

#define MAX 50

// Function to read a square matrix
void readMatrix(int arr[MAX][MAX], int n) {
    printf("Enter elements of %d x %d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &arr[i][j]);
        }
    }
}

// Function to display a square matrix
void displayMatrix(int arr[MAX][MAX], int n) {
    printf("\nThe %d x %d matrix is:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int arr[MAX][MAX];
    int n;

    printf("Enter size of square matrix (n): ");
    scanf("%d", &n);

    readMatrix(arr, n);      // function call to read
    displayMatrix(arr, n);   // function call to display

    return 0;
}
```

OUTPUT:

```
Enter size of square matrix (n): 3
Enter elements of 3 x 3 matrix:
1 2 3
4 5 6
7 8 9

The 3 x 3 matrix is:
1 2 3
4 5 6
7 8 9
```

11. Write a program that computes the sum of the elements that are stored on the main diagonal of a matrix using pointers.

```
#include <stdio.h>

#define MAX 50

int main() {
    int arr[MAX][MAX], n;
    int *ptr;   // pointer to traverse matrix
    int sum = 0;

    printf("Enter size of square matrix (n): ");
    scanf("%d", &n);

    printf("Enter elements of %d x %d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &arr[i][j]);
        }
    }

    // Pointer points to first element of matrix
    ptr = &arr[0][0];

    // Traverse diagonal elements using pointer arithmetic
    for (int i = 0; i < n; i++) {
        sum += *(ptr + i * n + i);  // accessing arr[i][i] using pointer
    }

    printf("\nSum of main diagonal elements = %d\n", sum);

    return 0;
}
```

OUTPUT:

```
Enter size of square matrix (n): 3
Enter elements of 3 x 3 matrix:
1 2 3
4 5 6
7 8 9

Sum of main diagonal elements = 15
```

12. Write a program to add two 3 * 3 matrix using pointers.

```
#include <stdio.h>

int main() {
    int A[3][3], B[3][3], C[3][3];
    int *pA, *pB, *pC;

    printf("Enter elements of first 3x3 matrix:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    printf("Enter elements of second 3x3 matrix:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            scanf("%d", &B[i][j]);
        }
    }

    // Initialize pointers
    pA = &A[0][0];
    pB = &B[0][0];
    pC = &C[0][0];

    // Add matrices using pointers
    for (int i = 0; i < 9; i++) {
        *(pC + i) = *(pA + i) + *(pB + i);
    }

    // Display result matrix
    printf("\nResultant Matrix (A + B):\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d\t", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT:

```
Enter elements of first 3x3 matrix:
1 2 3
4 5 6
7 8 9

Enter elements of second 3x3 matrix:
9 8 7
6 5 4
3 2 1

Resultant Matrix (A + B):
10    10    10
10    10    10
10    10    10
```

13. Write a program that computes the product of the elements that are stored on the diagonal above the main diagonal.

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the order of square matrix: ");
    scanf("%d", &n);

    int A[n][n];

    printf("Enter the elements of the %dx%d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    long long product = 1;
    int found = 0;  // To check if there are elements above diagonal

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            product *= A[i][j];
            found = 1;
        }
    }

    if (found)
        printf("\nProduct of elements above main diagonal = %lld\n", product);
    else
        printf("\nNo elements above main diagonal!\n");

    return 0;
}
```

OUTPUT:

```
Enter the order of square matrix: 3
Enter the elements of the 3x3 matrix:
1 2 3
4 5 6
7 8 9

Product of elements above main diagonal = 36
```

14. Write a program to count the total number of non-zero elements in a two-dimensional array.

```
#include <stdio.h>

int main() {
    int rows, cols;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int arr[rows][cols];

    // Input array elements
    printf("Enter the elements of the %dx%d matrix:\n", rows, cols);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &arr[i][j]);
        }
    }

    // Count non-zero elements
    int count = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (arr[i][j] != 0) {
                count++;
            }
        }
    }

    printf("\nTotal number of non-zero elements = %d\n", count);

    return 0;
}
```

OUTPUT:

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter the elements of the 3x3 matrix:
1 0 3
0 0 6
7 8 0

Total number of non-zero elements = 5
```

15. Write a program to input the elements of a two-dimensional array. Then from this array, make two arrays one that stores all odd elements of the two-dimensional array and the other that stores all even elements of the array.

```
#include <stdio.h>

int main() {
    int rows, cols;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    printf("Enter number of columns: ");
    scanf("%d", &cols);

    int arr[rows][cols];

    // Input elements
    printf("Enter the elements of the %dx%d matrix:\n", rows, cols);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &arr[i][j]);
        }
    }

    int even[rows * cols], odd[rows * cols];
    int eCount = 0, oCount = 0;

    // Separate odd and even
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (arr[i][j] % 2 == 0)
                even[eCount++] = arr[i][j];
            else
                odd[oCount++] = arr[i][j];
        }
    }

    // Display results
    printf("\nEven elements: ");
    if (eCount == 0)
        printf("None");
    else {
        for (int i = 0; i < eCount; i++)
            printf("%d ", even[i]);
    }

    printf("\nOdd elements: ");
    if (oCount == 0)
        printf("None");
    else {
        for (int i = 0; i < oCount; i++)
            printf("%d ", odd[i]);
    }

    printf("\n");
    return 0;
}
```

OUTPUT:

```
Enter number of rows: 3
Enter number of columns: 3
Enter the elements of the 3x3 matrix:
1 2 3
4 5 6
7 8 9

Even elements: 2 4 6 8
Odd elements: 1 3 5 7 9
```

16. Write a program to read two floating point number arrays. Merge the two arrays and display the resultant array in reverse order.

```
#include <stdio.h>

int main() {
    int n1, n2;

    printf("Enter size of first array: ");
    scanf("%d", &n1);
    printf("Enter size of second array: ");
    scanf("%d", &n2);

    float arr1[n1], arr2[n2], merged[n1 + n2];

    // Input first array
    printf("\nEnter %d elements of first array:\n", n1);
    for (int i = 0; i < n1; i++) {
        scanf("%f", &arr1[i]);
        merged[i] = arr1[i];  // copy directly
    }

    // Input second array
    printf("\nEnter %d elements of second array:\n", n2);
    for (int i = 0; i < n2; i++) {
        scanf("%f", &arr2[i]);
        merged[n1 + i] = arr2[i];  // append
    }

    int total = n1 + n2;

    // Display merged array in reverse
    printf("\nMerged array in reverse order:\n");
    for (int i = total - 1; i >= 0; i--) {
        printf("%.2f ", merged[i]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

```
Enter size of first array: 3
Enter size of second array: 2

Enter 3 elements of first array:
1.1 2.2 3.3

Enter 2 elements of second array:
4.4 5.5

Merged array in reverse order:
5.50 4.40 3.30 2.20 1.10
```

17. Write a program using pointers to interchange the second biggest and the second smallest number in the array.

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int *p = arr;

    // Initialize
    int smallest = *p, secondSmallest = 999999;
    int largest = *p, secondLargest = -999999;

    // Find smallest and largest
    for (int i = 0; i < n; i++) {
        if (*(p + i) < smallest)
            smallest = *(p + i);
        if (*(p + i) > largest)
            largest = *(p + i);
    }

    // Find second smallest and second largest
    for (int i = 0; i < n; i++) {
        if (*(p + i) > smallest && *(p + i) < secondSmallest)
            secondSmallest = *(p + i);
        if (*(p + i) < largest && *(p + i) > secondLargest)
            secondLargest = *(p + i);
    }

    // Interchange values in array
    for (int i = 0; i < n; i++) {
        if (*(p + i) == secondSmallest) {
            *(p + i) = secondLargest;
        } else if (*(p + i) == secondLargest) {
            *(p + i) = secondSmallest;
        }
    }

    // Print result
    printf("\nArray after interchanging second smallest (%d) and second largest
(%d):\n",
            secondSmallest, secondLargest);
    for (int i = 0; i < n; i++) {
        printf("%d ", *(p + i));
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

```
Enter size of array: 6
Enter 6 elements:
10 5 20 8 15 30

10 5 20 8 15 30   →   10 5 8 20 15 30

Array after interchanging second smallest (8) and second largest (20):
10 5 8 20 15 30
```

18. Write a menu driven program to read and display a pqr matrix. Also, find the sum, transpose, and product of the two pxqxr matrices.

```
#include <stdio.h>

int main() {
    int p, q, r, choice;

    // Read dimensions
    printf("Enter dimensions (p q r): ");
    scanf("%d %d %d", &p, &q, &r);

    int A[p][q], B[q][r], Sum[p][q], Trans[q][p], Product[p][r];

    // Input Matrix A
    printf("\nEnter elements of Matrix A (%dx%d):\n", p, q);
    for (int i = 0; i < p; i++)
        for (int j = 0; j < q; j++)
            scanf("%d", &A[i][j]);

    // Input Matrix B
    printf("\nEnter elements of Matrix B (%dx%d):\n", q, r);
    for (int i = 0; i < q; i++)
        for (int j = 0; j < r; j++)
            scanf("%d", &B[i][j]);

    do {
        printf("\n---- MENU ----\n");
        printf("1. Display Matrices\n");
        printf("2. Sum of A and B (only if p=q and q=r)\n");
        printf("3. Transpose of A\n");
        printf("4. Product of A and B\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: // Display matrices
                printf("\nMatrix A:\n");
                for (int i = 0; i < p; i++) {
```

```
        for (int j = 0; j < q; j++)
            printf("%d ", A[i][j]);
        printf("\n");
    }

    printf("\nMatrix B:\n");
    for (int i = 0; i < q; i++) {
        for (int j = 0; j < r; j++)
            printf("%d ", B[i][j]);
        printf("\n");
    }
    break;

case 2: // Sum (only possible if same order)
    if (p == q && q == r) {
        printf("\nSum of A and B:\n");
        for (int i = 0; i < p; i++) {
            for (int j = 0; j < q; j++) {
                Sum[i][j] = A[i][j] + B[i][j];
                printf("%d ", Sum[i][j]);
            }
            printf("\n");
        }
    } else {
        printf("\nSum not possible (different dimensions)\n");
    }
    break;

case 3: // Transpose of A
    printf("\nTranspose of A:\n");
    for (int i = 0; i < q; i++) {
        for (int j = 0; j < p; j++) {
            Trans[i][j] = A[j][i];
            printf("%d ", Trans[i][j]);
        }
        printf("\n");
    }
    break;

case 4: // Product A × B
```

```
            printf("\nProduct of A and B:\n");
            for (int i = 0; i < p; i++) {
                for (int j = 0; j < r; j++) {
                    Product[i][j] = 0;
                    for (int k = 0; k < q; k++)
                        Product[i][j] += A[i][k] * B[k][j];
                    printf("%d ", Product[i][j]);
                }
                printf("\n");
            }
            break;

        case 5:
            printf("Exiting program.\n");
            break;

        default:
            printf("Invalid choice!\n");
        }
    } while (choice != 5);

    return 0;
}
```

OUTPUT:

```
Enter dimensions (p q r): 2 2 2

Enter elements of Matrix A (2x2):
1 2
3 4

Enter elements of Matrix B (2x2):
5 6
7 8

// Sum
6 8
10 12

// Transpose A
1 3
2 4

// Product (A * B)
19 22
43 50
```

19.Write a program that reads a matrix and displays the sum of its diagonal elements.

```
#include <stdio.h>

int main() {
   int n;
   printf("Enter order of square matrix (n x n): ");
   scanf("%d", &n);

   int arr[n][n];
   int sum = 0;

   // Input matrix
   printf("Enter elements of %dx%d matrix:\n", n, n);
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &arr[i][j]);
         if (i == j) {


            sum += arr[i][j];  // add diagonal element
         }
      }
   }

   // Display matrix
   printf("\nMatrix:\n");
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }

   // Display result
   printf("\nSum of diagonal elements = %d\n", sum);

   return 0;
}
```

OUTPUT:

```
Enter order of square matrix (n x n): 3
Enter elements of 3x3 matrix:
1 2 3
4 5 6
7 8 9

Matrix:
1 2 3
4 5 6
7 8 9

Sum of diagonal elements = 15
```

20. Write a program that reads a matrix and displays the sum of the elements above the main diagonal. (Hint: Calculate the sum of elements A ij where icf).

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter order of square matrix (n x n): ");
    scanf("%d", &n);

    int arr[n][n];
    int sum = 0;

    // Input matrix
    printf("Enter elements of %dx%d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &arr[i][j]);
            if (j > i) {   // element above main diagonal
                sum += arr[i][j];
            }
        }
    }

    // Display matrix
    printf("\nMatrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }

    // Display result
    printf("\nSum of elements above main diagonal = %d\n", sum);

    return 0;
}
```

OUTPUT:

```
Enter order of square matrix (n x n): 3
Enter elements of 3x3 matrix:
1 2 3
4 5 6
7 8 9

Matrix:
1 2 3
4 5 6
7 8 9

Sum of elements above main diagonal = 11
```

21. Write a program that reads a matrix displays the sum of the elements below the main diagonal. (Hint: Calculate the sum of elements A_{G} where (sj)

```c
#include <stdio.h>

int main() {
    int n;
    printf("Enter order of square matrix (n x n): ");
    scanf("%d", &n);

    int arr[n][n];
    int sum = 0;

    // Input matrix
    printf("Enter elements of %dx%d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &arr[i][j]);
            if (i > j) {   // elements below main diagonal
                sum += arr[i][j];
            }
        }
    }

    // Display matrix
    printf("\nMatrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }

    // Display result
    printf("\nSum of elements below main diagonal = %d\n", sum);

    return 0;
}
```

OUTPUT:

```
Enter order of square matrix (n x n): 3
Enter elements of 3x3 matrix:
1 2 3
4 5 6
7 8 9

Matrix:
1 2 3
4 5 6
7 8 9

Sum of elements below main diagonal = 19
```

22. Write a program that reads a square matrix of size nxn. Write a function int isUpperTriangular (int a[][], int n) that returns 1 if the matrix is upper triangular. (Hint: Array A is upper triangular if A ij =0 and)

A[i][j] == 0 for all i > j

```
1  2  3
0  4  5
0  0  6
```

```
1  2  3
4  5  6   ← 4 is below diagonal (not zero)
0  0  7
```

```
#include <stdio.h>

// Function to check if matrix is upper triangular
int isUpperTriangular(int a[20][20], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) { // check only below main diagonal
            if (a[i][j] != 0) {
                return 0; // Not upper triangular
            }
        }
    }
    return 1; // It is upper triangular
}

int main() {
    int n, a[20][20];
    printf("Enter order of square matrix (n x n): ");
    scanf("%d", &n);

    printf("Enter elements of %dx%d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    // Display matrix
    printf("\nMatrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }

    // Check
    if (isUpperTriangular(a, n))
        printf("\nThe matrix is Upper Triangular.\n");
    else
        printf("\nThe matrix is NOT Upper Triangular.\n");

    return 0;
}
```

OUTPUT:

```
Enter order of square matrix (n x n): 3
Enter elements of 3x3 matrix:
1 2 3
0 4 5
0 0 6

Matrix:
1 2 3
0 4 5
0 0 6

The matrix is Upper Triangular.
```

23. Write a program that reads a square matrix of size nxn. Write a function int isLowerTriangular (int a[][], int n) that returns I if the matrix is lower triangular. (Hint: Array A is lower triangular d^ prime A ij = theta and <j).

A[i][j] == 0 for all j > i

1 0 0
4 5 0
7 8 9

1 2 0 ← 2 is above diagonal (not zero)
4 5 0
7 8 9

```
#include <stdio.h>

// Function to check if matrix is lower triangular
int isLowerTriangular(int a[20][20], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) { // check only above main diagonal
            if (a[i][j] != 0) {
                return 0; // Not lower triangular
            }
        }
    }
    return 1; // It is lower triangular
}

int main() {
    int n, a[20][20];
    printf("Enter order of square matrix (n x n): ");
    scanf("%d", &n);

    printf("Enter elements of %dx%d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    // Display matrix
    printf("\nMatrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }

    // Check
    if (isLowerTriangular(a, n))
        printf("\nThe matrix is Lower Triangular.\n");
    else
        printf("\nThe matrix is NOT Lower Triangular.\n");

    return 0;
}
```

OUTPUT:

```
Enter order of square matrix (n x n): 3
Enter elements of 3x3 matrix:
1 0 0
4 5 0
7 8 9

Matrix:
1 0 0
4 5 0
7 8 9

The matrix is Lower Triangular.
```

24. Write a program that reads a square matrix of size nxn. Write a function int isSymmetric (int \mathfrak{a} [][], int n) that returns 1 if the matrix is symmetric. (Hint: Array A is symmetric if A \mathfrak{A}_{p}*for all values of i and 5).

Symmetric Example:

1 2 3
2 4 5
3 5 6

Non - Symmetric Example:

1 0 2
0 4 5
9 5 6 ← A[0][2] = 2 but A[2][0] = 9

```
#include <stdio.h>

// Function to check if matrix is symmetric
int isSymmetric(int a[20][20], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i][j] != a[j][i]) {
                return 0; // Not symmetric
            }
        }
    }
    return 1; // Symmetric
}

int main() {
    int n, a[20][20];
    printf("Enter order of square matrix (n x n): ");
    scanf("%d", &n);

    printf("Enter elements of %dx%d matrix:\n", n, n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    // Display matrix
    printf("\nMatrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }

    // Check
    if (isSymmetric(a, n))
        printf("\nThe matrix is Symmetric.\n");
    else
        printf("\nThe matrix is NOT Symmetric.\n");

    return 0;
}
```

OUTPUT:

```
Enter order of square matrix (n x n): 3
Enter elements of 3x3 matrix:
1 2 3
2 4 5
3 5 6

Matrix:
1 2 3
2 4 5
3 5 6

The matrix is Symmetric.
```

25. Write a program to calculate XA + YB where Aand B are matrices and X = 2 and Y = 3

```
#include <stdio.h>

int main() {
    int p, q;
    int X = 2, Y = 3;

    printf("Enter number of rows: ");
    scanf("%d", &p);
    printf("Enter number of columns: ");
    scanf("%d", &q);

    int A[p][q], B[p][q], Result[p][q];

    // Input Matrix A
    printf("\nEnter elements of Matrix A (%dx%d):\n", p, q);
    for (int i = 0; i < p; i++)
        for (int j = 0; j < q; j++)
            scanf("%d", &A[i][j]);

    // Input Matrix B
    printf("\nEnter elements of Matrix B (%dx%d):\n", p, q);
    for (int i = 0; i < p; i++)
        for (int j = 0; j < q; j++)
            scanf("%d", &B[i][j]);

    // Compute XA + YB
    for (int i = 0; i < p; i++) {
        for (int j = 0; j < q; j++) {
            Result[i][j] = X * A[i][j] + Y * B[i][j];
        }
    }

    // Display result
    printf("\nMatrix XA + YB:\n");
    for (int i = 0; i < p; i++) {
        for (int j = 0; j < q; j++)
            printf("%d ", Result[i][j]);
        printf("\n");
    }

    return 0;
}
```

OUTPUT:

```
Enter number of rows: 2
Enter number of columns: 2

Enter elements of Matrix A (2x2):
1 2
3 4

Enter elements of Matrix B (2x2):
5 6
7 8

Matrix XA + YB:
17 22
29 34
```

26. Write a program to illustrate the use of a pointer that points to a 2D array.

int (*ptr)[COLS] = arr;

```
#include <stdio.h>

int main() {
    int rows, cols;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    printf("Enter number of columns: ");
    scanf("%d", &cols);

    int arr[rows][cols];

    // Input elements
    printf("Enter elements of %dx%d matrix:\n", rows, cols);
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            scanf("%d", &arr[i][j]);

    // Pointer to 2D array
    int (*ptr)[cols] = arr;

    // Display elements using pointer
    printf("\nMatrix elements using pointer:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", ptr[i][j]);
        }
        printf("\n");
    }

    // Example: Access a specific element
    printf("\nElement at [1][1] using pointer = %d\n", ptr[1][1]);

    return 0;
}
```

OUTPUT:

```
Enter number of rows: 2
Enter number of columns: 3
Enter elements of 2x3 matrix:
1 2 3
4 5 6

Matrix elements using pointer:
1 2 3
4 5 6

Element at [1][1] using pointer = 5
```

27. Write a program to enter a number and break it into a number of digits.

```c
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    int original = num;

    printf("Digits of %d are: ", original);

    // To store digits in correct order, first count digits
    int digits[10], i = 0;

    while (num > 0) {
        digits[i] = num % 10; // extract last digit
        num = num / 10;       // remove last digit
        i++;
    }

    // Print digits in correct order
    for (int j = i - 1; j >= 0; j--) {
        printf("%d ", digits[j]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

```
Enter a number: 4729

Digits of 4729 are: 4 7 2 9
```

28.Write a program to delete all the duplicate entries from an array of n integers.

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // Remove duplicates
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; ) {
            if (arr[i] == arr[j]) {
                // Shift elements left
                for (int k = j; k < n - 1; k++) {
                    arr[k] = arr[k + 1];
                }
                n--; // Reduce array size
            } else {
                j++;
            }
        }
    }

    // Display array after removing duplicates
    printf("\nArray after removing duplicates:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

```
Enter the number of elements: 8
Enter 8 elements:
1 2 3 2 4 1 5 3

Array after removing duplicates:
1 2 3 4 5
```

29.Write a program to read a floating point array.

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    float arr[n];

    // Input array elements
    printf("Enter %d floating-point numbers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%f", &arr[i]);
    }

    // Display array elements
    printf("\nThe elements of the array are:\n");
    for (int i = 0; i < n; i++) {
        printf("%.2f ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

```
Enter the number of elements in the array: 5
Enter 5 floating-point numbers:
1.2 3.4 5.6 7.8 9.0

The elements of the array are:
1.20 3.40 5.60 7.80 9.00
```