# LAB 08 TASKS

**Question # 1)** Read the entries of an array of 10 integers from a user. Compute x as the average of the 10 entries and then compute the average and display those entries that are greater than or equal to x. Print this final average.

**Source Code:**

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int arr[10];
    double sum = 0, x;

    // Input 10 integers from the user
    cout << "Enter 10 integers: \n";
    for (int i = 0; i < 10; ++i) {
        cin >> arr[i];
        sum += arr[i];
    }

    // Compute the average (x) of the 10 integers
    x = sum / 10;
    cout << "The average of the 10 entries is: " << x << endl;

    // Find entries greater than or equal to x
    vector<int> greater_or_equal;
    sum = 0; // Reset sum for the new average
    for (int i = 0; i < 10; ++i) {
        if (arr[i] >= x) {
            greater_or_equal.push_back(arr[i]);
            sum += arr[i];
        }
    }

    // Compute the average of the selected entries
    double final_average = 0;
    if (!greater_or_equal.empty()) {
        final_average = sum / greater_or_equal.size();
    }

    // Display the selected entries and their average
    cout << "Entries greater than or equal to the average: ";
    for (int num : greater_or_equal) {
        cout << num << " ";
    }
    cout << "\nThe average of these entries is: " << final_average << endl;

    return 0;
}
```

**Output:**

```
Enter 10 integers:
1
2
3
4
5
6
7
8
9
10
The average of the 10 entries is: 5.5
Entries greater than or equal to the average: 6 7 8 9 10
The average of these entries is: 8

--------------------------------
Process exited after 10.86 seconds with return value 0
Press any key to continue . . .
```

**Question # 2)** Write a C++ code to find the minimum and maximum distance between two numbers of an array.

**Source Code:**

```cpp
#include <iostream>
#include <cmath>
#include <limits>
using namespace std;

int main() {
    int n, num1, num2;

    // Input the size of the array
    cout << "Enter the size of the array: ";
    cin >> n;

    if (n < 2) {
        cout << "Array must have at least two elements." << endl;
        return 0;
    }

    int arr[n];

    // Input the array elements
    cout << "Enter " << n << " elements: \n";
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }

    // Input the two numbers to find distances between
    cout << "Enter the two numbers to find distances: \n";
    cin >> num1 >> num2;

    int min_distance = numeric_limits<int>::max();
    int max_distance = -1;
    int last_position_num1 = -1;
    int last_position_num2 = -1;

    // Traverse the array to find minimum and maximum distances
    for (int i = 0; i < n; ++i) {
        if (arr[i] == num1) {
            if (last_position_num2 != -1) {
                int distance = abs(i - last_position_num2);
                min_distance = min(min_distance, distance);
                max_distance = max(max_distance, distance);
            }
            last_position_num1 = i;
        } else if (arr[i] == num2) {
            if (last_position_num1 != -1) {
                int distance = abs(i - last_position_num1);
                min_distance = min(min_distance, distance);
                max_distance = max(max_distance, distance);
            }
            last_position_num2 = i;
        }
    }

    if (min_distance == numeric_limits<int>::max()) {
        cout << "The two numbers do not both appear in the array." << endl;
    } else {
        cout << "The minimum distance between " << num1 << " and " << num2 << " is: " << min_distance << endl;
        cout << "The maximum distance between " << num1 << " and " << num2 << " is: " << max_distance << endl;
    }

    return 0;
}
```

**Output:**

```
Enter the size of the array: 3
Enter 3 elements:
23
24
25
Enter the two numbers to find distances:
23
25
The minimum distance between 23 and 25 is: 2
The maximum distance between 23 and 25 is: 2

--------------------------------
Process exited after 16.75 seconds with return value 0
Press any key to continue . . .
```

**Question # 3)** Take input 10 numbers from user, sort them in ascending and descending order.

**Source Code:**

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    const int SIZE = 10;
    int arr[SIZE];

    // Input 10 numbers from the user
    cout << "Enter 10 numbers: \n";
    for (int i = 0; i < SIZE; ++i) {
        cin >> arr[i];
    }

    // Sort the array in ascending order
    sort(arr, arr + SIZE);

    // Display the sorted array in ascending order
    cout << "Numbers in ascending order: \n";
    for (int i = 0; i < SIZE; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;

    // Sort the array in descending order
    sort(arr, arr + SIZE, greater<int>());

    // Display the sorted array in descending order
    cout << "Numbers in descending order: \n";
    for (int i = 0; i < SIZE; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

**Output:**

```
Enter 10 numbers:
9
8
7
6
5
4
5
3
2
1
Numbers in ascending order:
1 2 3 4 5 5 6 7 8 9
Numbers in descending order:
9 8 7 6 5 5 4 3 2 1

--------------------------------
Process exited after 13.44 seconds with return value 0
Press any key to continue . . .
```

**Question # 4)** Take array of 5 numbers from user, now print them in reverse order.

**Source Code:**

```cpp
#include <iostream>
using namespace std;

int main() {
    const int SIZE = 5;
    int arr[SIZE];

    // Input 5 numbers from the user
    cout << "Enter 5 numbers: \n";
    for (int i = 0; i < SIZE; ++i) {
        cin >> arr[i];
    }

    // Display the array in reverse order
    cout << "Numbers in reverse order: \n";
    for (int i = SIZE - 1; i >= 0; --i) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

**Output:**

```
Enter 5 numbers:
5
6
7
8
9
Numbers in reverse order:
9 8 7 6 5

--------------------------------
Process exited after 8.948 seconds with return value 0
Press any key to continue . . .
```

**Question # 5)** Take 10 float numbers from user, now find second greatest number from array.

**Source Code:**

```cpp
#include <iostream>
#include <limits>
using namespace std;

int main() {
    float numbers[10];
    float greatest = -numeric_limits<float>::infinity();
    float secondGreatest = -numeric_limits<float>::infinity();

    // Taking 10 float numbers from the user
    cout << "Enter 10 float numbers:" << endl;
    for (int i = 0; i < 10; i++) {
        cin >> numbers[i];

        // Update greatest and second greatest numbers
        if (numbers[i] > greatest) {
            secondGreatest = greatest;
            greatest = numbers[i];
        } else if (numbers[i] > secondGreatest && numbers[i] != greatest) {
            secondGreatest = numbers[i];
        }
    }

    if (secondGreatest == -numeric_limits<float>::infinity()) {
        cout << "There is no distinct second greatest number." << endl;
    } else {
        cout << "The second greatest number is: " << secondGreatest << endl;
    }

    return 0;
}
```

**Output:**

```
Enter 10 float numbers:
1.1
2.2
3.3
4.4
5.5
6.6
7.7
8.8
9.9
10.10
The second greatest number is: 9.9

--------------------------------
Process exited after 41.55 seconds with return value 0
Press any key to continue . . .
```

**Question # 6)** Take array of 10 numbers, now find smallest number in array and make it the greatest number in array and then print new array.

**Source Code:**

```cpp
#include <iostream>
#include <limits>
using namespace std;

int main() {
    float numbers[10];
    float smallest;

    // Taking input from the user
    cout << "Enter 10 float numbers: \n";
    for(int i = 0; i < 10; i++) {
        cin >> numbers[i];
    }

    // Finding the smallest number in the array
    smallest = numbers[0];
    for(int i = 1; i < 10; i++) {
        if(numbers[i] < smallest) {
            smallest = numbers[i];
        }
    }

    // Making the smallest number the greatest
    for(int i = 0; i < 10; i++) {
        if(numbers[i] == smallest) {
            numbers[i] = numeric_limits<float>::infinity();  // Temporary set to infinity
        }
    }

    // Replacing the smallest (temporary infinity) with the greatest value
    for(int i = 0; i < 10; i++) {
        if(numbers[i] == numeric_limits<float>::infinity()) {
            numbers[i] = smallest;
        }
    }

    // Printing the modified array
    cout << "Modified array with smallest number as the greatest: \n";
    for(int i = 0; i < 10; i++) {
        cout << numbers[i] << " ";
    }
    cout << endl;

    return 0;
}
```

**Output:**

```
Enter 10 float numbers:
1.1
2.2
3.3
4.4
5.5
6.6
7.7
8.8
9.9
10.10
Modified array with smallest number as the greatest:
1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9 10.1

--------------------------------
Process exited after 27.72 seconds with return value 0
Press any key to continue . . .
```

**Question # 7)** Take 10 numbers from user, now display most occurring element and also its number of occurrence.

**Source Code:**

```cpp
#include <iostream>
using namespace std;

int main() {
    int numbers[10];
    int mostOccurring, maxCount = 0;

    // Taking input from the user
    cout << "Enter 10 numbers: \n";
    for (int i = 0; i < 10; i++) {
        cin >> numbers[i];
    }

    // Find the most occurring element
    for (int i = 0; i < 10; i++) {
        int count = 0;
        for (int j = 0; j < 10; j++) {
            if (numbers[i] == numbers[j]) {
                count++;
            }
        }
        // Update if this number occurs more times than the previous one
        if (count > maxCount) {
            maxCount = count;
            mostOccurring = numbers[i];
        }
    }

    // Output the result
    cout << "Most occurring element: " << mostOccurring << endl;
    cout << "Number of occurrences: " << maxCount << endl;

    return 0;
}
```

**Output:**

```
Enter 10 numbers:
5
2
1
5
4
2
1
5
3
6
Most occurring element: 5
Number of occurrences: 3

--------------------------------
Process exited after 8.836 seconds with return value 0
Press any key to continue . . .
```

**Question # 8)** Write a C++ program to generate the sum of left diagonal.

**Source Code:**

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;

    // Taking input for the size of the matrix
    cout << "Enter the size of the matrix (n x n): ";
    cin >> n;

    int matrix[n][n];
    int leftDiagonalSum = 0;

    // Taking input for the matrix elements
    cout << "Enter the elements of the matrix: \n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }

    // Calculating the sum of the left diagonal
    for (int i = 0; i < n; i++) {
        leftDiagonalSum += matrix[i][i]; // Adding elements where row index equals column index
    }

    // Outputting the result
    cout << "Sum of the left diagonal: " << leftDiagonalSum << endl;

    return 0;
}
```

**Output:**

```
Enter the size of the matrix (n x n): 3
Enter the elements of the matrix:
1
2
3
4
5
6
7
8
9
Sum of the left diagonal: 15

--------------------------------
Process exited after 13.35 seconds with return value 0
Press any key to continue . . .
```

**Question # 9)** Write a C++ program to find the duplicate values in a 2d array.

**Source Code:**

```cpp
#include <iostream>
using namespace std;

int main() {
    int rows, cols;

    // Taking input for the size of the 2D array
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> cols;

    int array[rows][cols];

    // Taking input for the 2D array elements
    cout << "Enter the elements of the 2D array: \n";
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cin >> array[i][j];
        }
    }

    bool foundDuplicate = false;

    // Finding and printing duplicate values
    cout << "Duplicate values in the array: \n";
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            for (int k = i; k < rows; k++) {
                for (int l = (k == i) ? j + 1 : 0; l < cols; l++) {
                    if (array[i][j] == array[k][l]) {
                        cout << array[i][j] << " ";
                        foundDuplicate = true;
                        break;
                    }
                }
            }
        }
    }

    if (!foundDuplicate) {
        cout << "No duplicate values found." << endl;
    }

    return 0;
}
```

**Output:**

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter the elements of the 2D array:
2
5
41
5
2
3
6
45
5
Duplicate values in the array:
2 5 5 5
--------------------------------
Process exited after 12.87 seconds with return value 0
Press any key to continue . . .
```

**Question # 10)** Write a C++ program to move all negative elements of an array of integers to the end of the array without changing the order of positive element and negative element.

**Source Code:**

```cpp
#include <iostream>
using namespace std;

void moveNegativesToEnd(int arr[], int n) {
    int positiveIndex = 0;

    // Move all positive numbers to the beginning of the array
    for (int i = 0; i < n; i++) {
        if (arr[i] >= 0) {
            arr[positiveIndex++] = arr[i];
        }
    }

    // Fill the rest of the array with negative numbers
    for (int i = 0; i < n; i++) {
        if (arr[i] < 0) {
            arr[positiveIndex++] = arr[i];
        }
    }
}

int main() {
    int n;

    // Input the size of the array
    cout << "Enter the number of elements in the array: ";
    cin >> n;

    int arr[n];

    // Input array elements
    cout << "Enter the elements of the array: \n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Move negative elements to the end
    moveNegativesToEnd(arr, n);

    // Output the modified array
    cout << "Array after moving negative elements to the end: \n";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

**Output:**

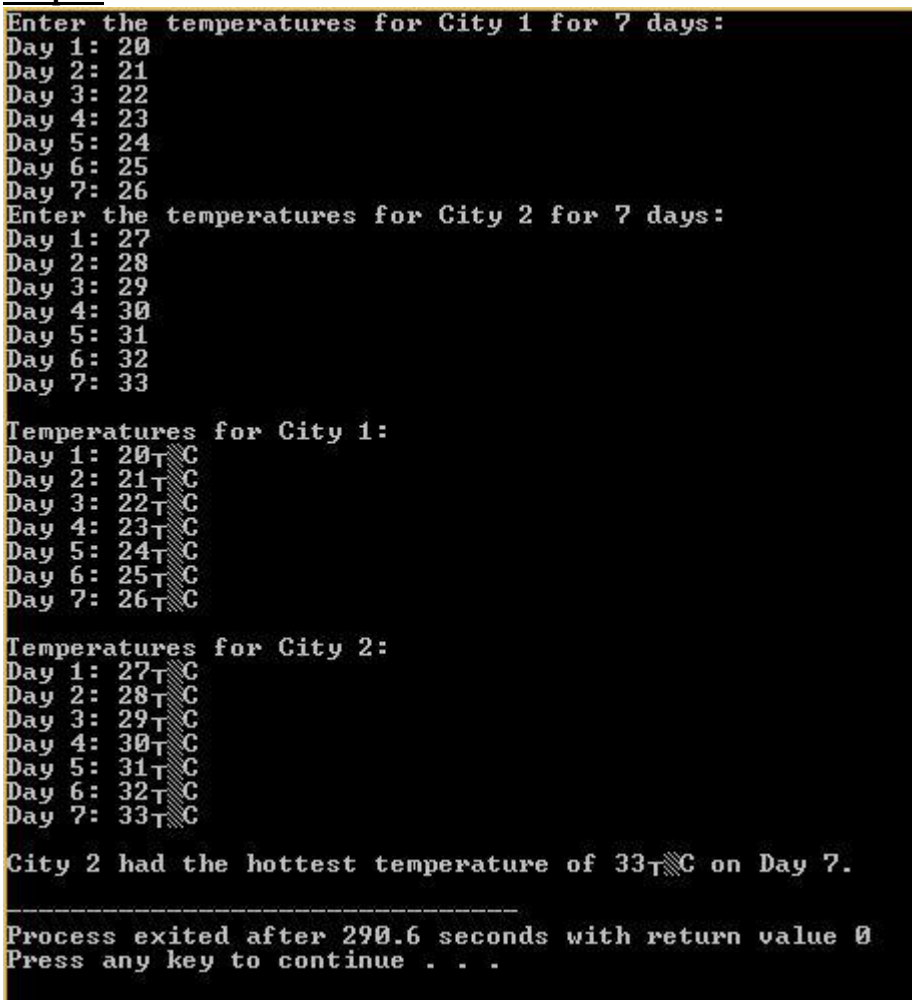
```
Enter the number of elements in the array: 4
Enter the elements of the array:
9
8
7
6
Array after moving negative elements to the end:
9 8 7 6

--------------------------------
Process exited after 7.327 seconds with return value 0
Press any key to continue . . .
```

**Question # 11)** Write a C++ Program to store temperature of two different cities for a week and display it. Find the city with hottest temperature.

**Source Code:**

```cpp
#include <iostream>
using namespace std;

int main() {
    const int daysInWeek = 7;
    float city1Temps[daysInWeek], city2Temps[daysInWeek];
    float city1MaxTemp = -1000, city2MaxTemp = -1000; // Initializing with a very low value
    int city1MaxDay = 0, city2MaxDay = 0;

    // Taking input for the temperatures of city 1 for the week
    cout << "Enter the temperatures for City 1 for 7 days:\n";
    for (int i = 0; i < daysInWeek; i++) {
        cout << "Day " << i + 1 << ": ";
        cin >> city1Temps[i];
    }

    // Taking input for the temperatures of city 2 for the week
    cout << "Enter the temperatures for City 2 for 7 days:\n";
    for (int i = 0; i < daysInWeek; i++) {
        cout << "Day " << i + 1 << ": ";
        cin >> city2Temps[i];
    }

    // Displaying the temperatures for both cities
    cout << "\nTemperatures for City 1:\n";
    for (int i = 0; i < daysInWeek; i++) {
        cout << "Day " << i + 1 << ": " << city1Temps[i] << "°C\n";
    }

    cout << "\nTemperatures for City 2:\n";
    for (int i = 0; i < daysInWeek; i++) {
        cout << "Day " << i + 1 << ": " << city2Temps[i] << "°C\n";
    }

    // Finding the hottest day for City 1
    for (int i = 0; i < daysInWeek; i++) {
        if (city1Temps[i] > city1MaxTemp) {
            city1MaxTemp = city1Temps[i];
            city1MaxDay = i + 1; // Day of the hottest temperature
        }
    }

    // Finding the hottest day for City 2
    for (int i = 0; i < daysInWeek; i++) {
        if (city2Temps[i] > city2MaxTemp) {
            city2MaxTemp = city2Temps[i];
            city2MaxDay = i + 1; // Day of the hottest temperature
        }
    }

    // Comparing which city has the hottest temperature
    if (city1MaxTemp > city2MaxTemp) {
        cout << "\nCity 1 had the hottest temperature of " << city1MaxTemp << "°C on Day " << city1MaxDay << ".\n";
    } else if (city2MaxTemp > city1MaxTemp) {
        cout << "\nCity 2 had the hottest temperature of " << city2MaxTemp << "°C on Day " << city2MaxDay << ".\n";
    } else {
        cout << "\nBoth cities had the same hottest temperature of " << city1MaxTemp << "°C.\n";
    }

    return 0;
}
```

**Output:**


Enter the temperatures for City 1 for 7 days:
Day 1: 20
Day 2: 21
Day 3: 22
Day 4: 23
Day 5: 24
Day 6: 25
Day 7: 26
Enter the temperatures for City 2 for 7 days:
Day 1: 27
Day 2: 28
Day 3: 29
Day 4: 30
Day 5: 31
Day 6: 32
Day 7: 33
Temperatures for City 1:
Day 1: 20°C
Day 2: 21°C
Day 3: 22°C
Day 4: 23°C
Day 5: 24°C
Day 6: 25°C
Day 7: 26°C
Temperatures for City 2:
Day 1: 27°C
Day 2: 28°C
Day 3: 29°C
Day 4: 30°C
Day 5: 31°C
Day 6: 32°C
Day 7: 33°C
City 2 had the hottest temperature of 33°C on Day 7.
Process exited after 290.6 seconds with return value 0
Press any key to continue . . .

**Question # 12)** Write a C++ program to generate transpose of 3*3 matrix

**Source Code:**

```cpp
#include <iostream>
using namespace std;

int main() {
    int matrix[3][3], transpose[3][3];

    // Taking input for the 3x3 matrix
    cout << "Enter the elements of the 3x3 matrix:\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "Element at position (" << i + 1 << "," << j + 1 << "): ";
            cin >> matrix[i][j];
        }
    }

    // Generating the transpose of the matrix
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            transpose[i][j] = matrix[j][i];  // Transpose: swap rows and columns
        }
    }

    // Displaying the original matrix
    cout << "\nOriginal Matrix:\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }

    // Displaying the transposed matrix
    cout << "\nTranspose of the Matrix:\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << transpose[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

**Output:**

```
Enter the elements of the 3x3 matrix:
Element at position (1,1): 3
Element at position (1,2): 4
Element at position (1,3): 5
Element at position (2,1): 6
Element at position (2,2): 9
Element at position (2,3): 8
Element at position (3,1): 1
Element at position (3,2): 5
Element at position (3,3): 0

Original Matrix:
3 4 5
6 9 8
1 5 0

Transpose of the Matrix:
3 6 1
4 9 5
5 8 0

--------------------------------
Process exited after 24.41 seconds with return value 0
Press any key to continue . . .
```