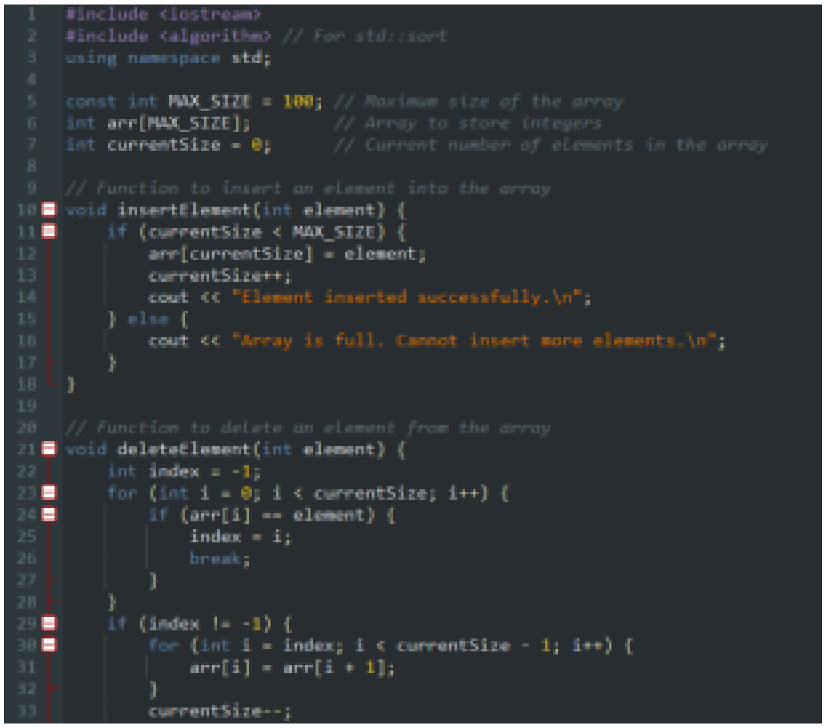
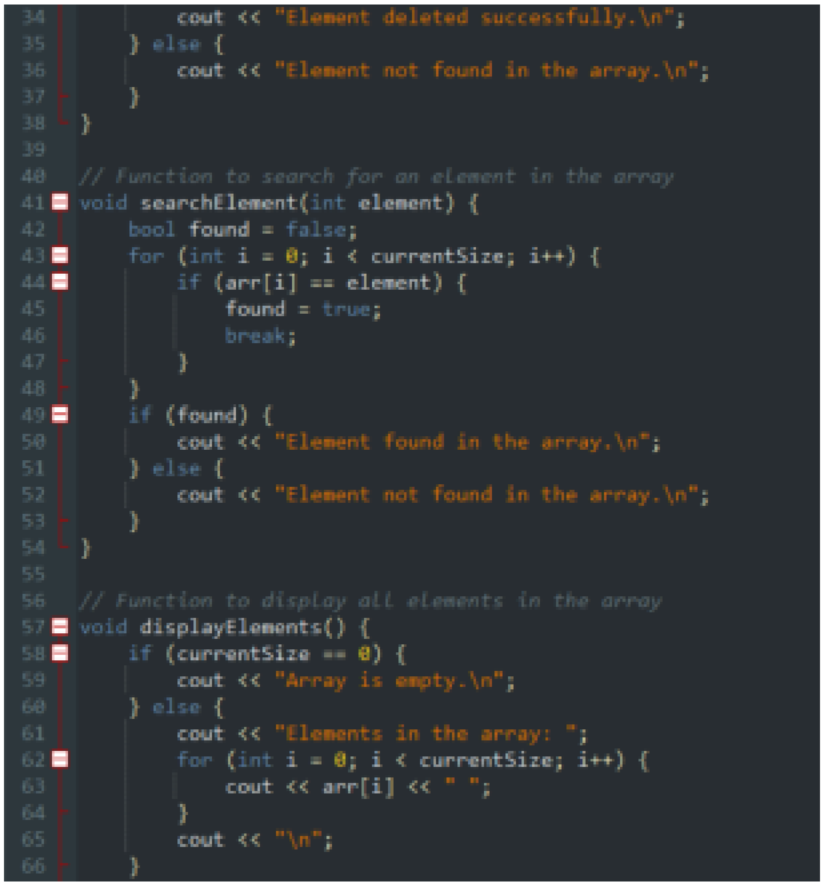
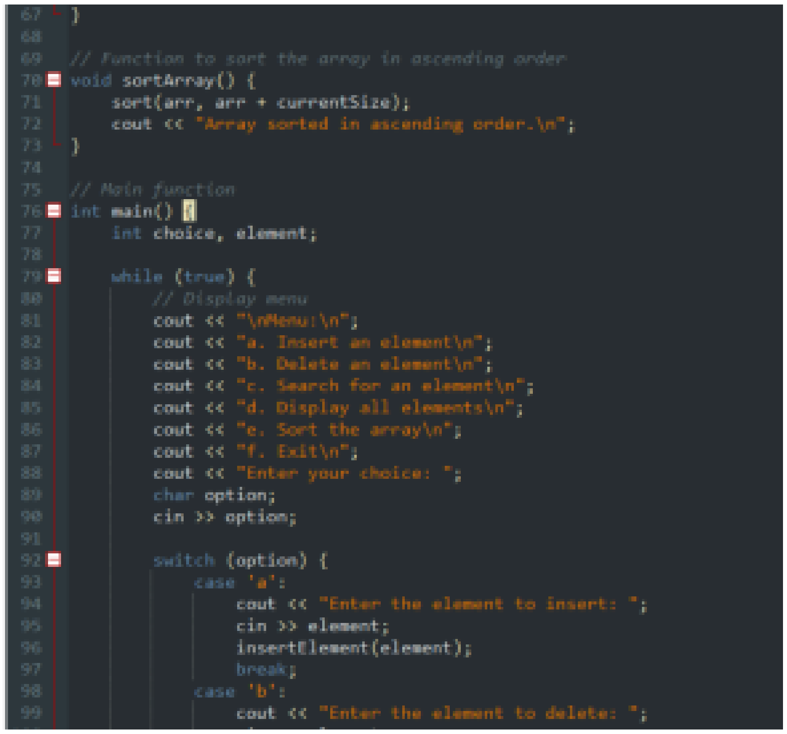
PROBLEM-1



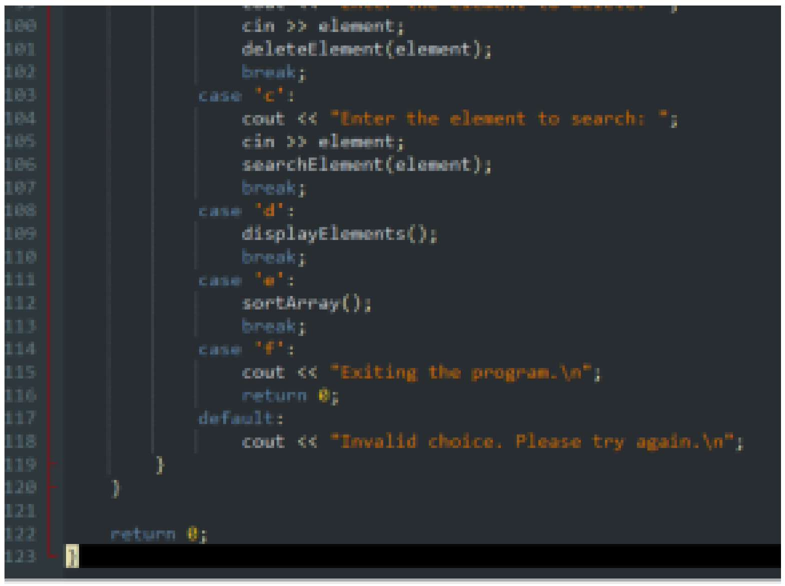
```cpp
#include <iostream>
#include <algorithm> // For std::sort
using namespace std;

const int MAX_SIZE = 100; // Maximum size of the array
int arr[MAX_SIZE];         // Array to store integers
int currentSize = 0;       // Current number of elements in the array

// Function to insert an element into the array
void insertElement(int element) {
    if (currentSize < MAX_SIZE) {
        arr[currentSize] = element;
        currentSize++;
        cout << "Element inserted successfully.\n";
    } else {
        cout << "Array is full. Cannot insert more elements.\n";
    }
}

// Function to delete an element from the array
void deleteElement(int element) {
    int index = -1;
    for (int i = 0; i < currentSize; i++) {
        if (arr[i] == element) {
            index = i;
            break;
        }
    }
    if (index != -1) {
        for (int i = index; i < currentSize - 1; i++) {
            arr[i] = arr[i + 1];
        }
        currentSize--;
```



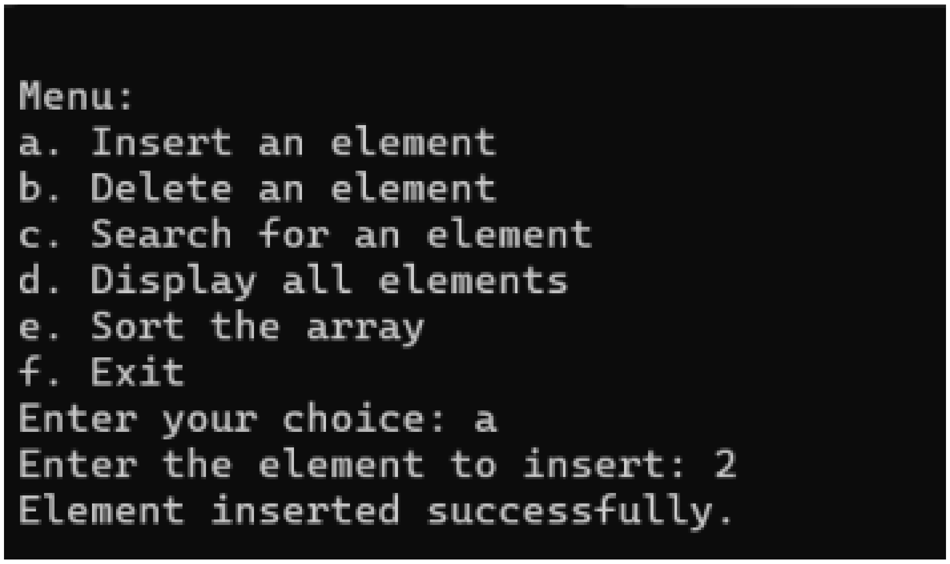
```cpp
        cout << "Element deleted successfully.\n";
    } else {
        cout << "Element not found in the array.\n";
    }
}

// Function to search for an element in the array
void searchElement(int element) {
    bool found = false;
    for (int i = 0; i < currentSize; i++) {
        if (arr[i] == element) {
            found = true;
            break;
        }
    }
    if (found) {
        cout << "Element found in the array.\n";
    } else {
        cout << "Element not found in the array.\n";
    }
}

// Function to display all elements in the array
void displayElements() {
    if (currentSize == 0) {
        cout << "Array is empty.\n";
    } else {
        cout << "Elements in the array: ";
        for (int i = 0; i < currentSize; i++) {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
```



```cpp
}

// Function to sort the array in ascending order
void sortArray() {
    sort(arr, arr + currentSize);
    cout << "Array sorted in ascending order.\n";
}

// Main function
int main() {
    int choice, element;

    while (true) {
        // Display menu
        cout << "\nMenu:\n";
        cout << "a. Insert an element\n";
        cout << "b. Delete an element\n";
        cout << "c. Search for an element\n";
        cout << "d. Display all elements\n";
        cout << "e. Sort the array\n";
        cout << "f. Exit\n";
        cout << "Enter your choice: ";
        char option;
        cin >> option;

        switch (option) {
            case 'a':
                cout << "Enter the element to insert: ";
                cin >> element;
                insertElement(element);
                break;
            case 'b':
                cout << "Enter the element to delete: ";
```

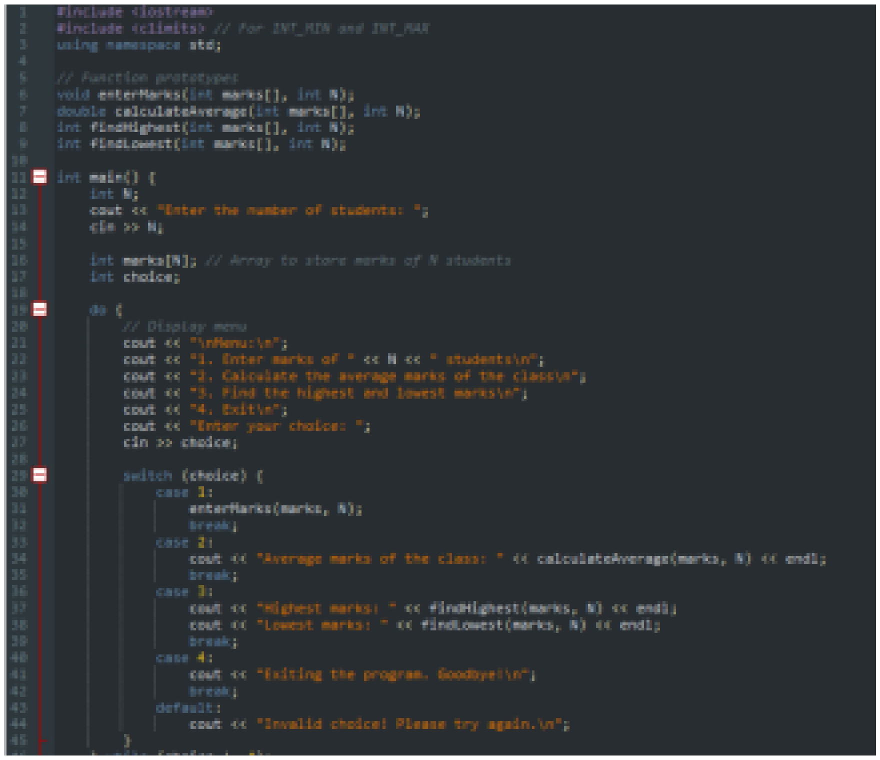


```cpp
                cin >> element;
                deleteElement(element);
                break;
            case 'c':
                cout << "Enter the element to search: ";
                cin >> element;
                searchElement(element);
                break;
            case 'd':
                displayElements();
                break;
            case 'e':
                sortArray();
                break;
            case 'f':
                cout << "Exiting the program.\n";
                return 0;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    }

    return 0;
}
```

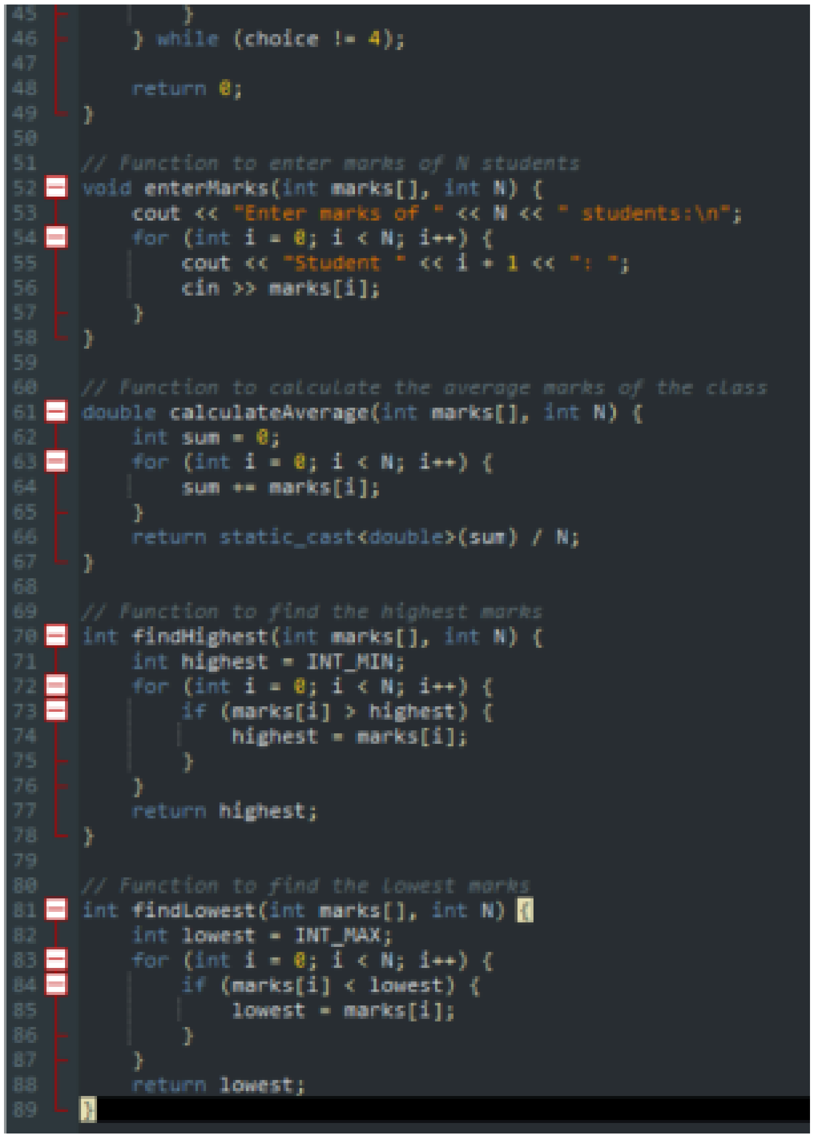


```
Menu:
a. Insert an element
b. Delete an element
c. Search for an element
d. Display all elements
e. Sort the array
f. Exit
Enter your choice: a
Enter the element to insert: 2
Element inserted successfully.
```

PROBLEM-2

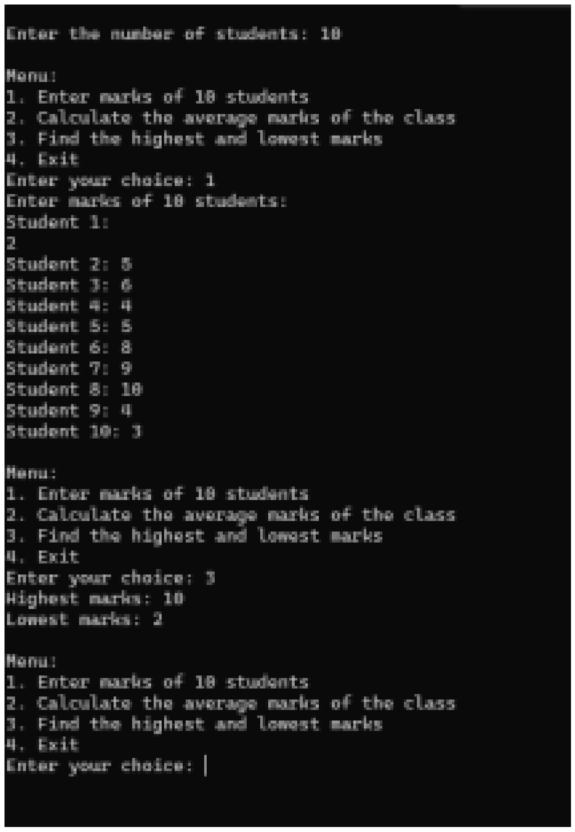


```cpp
#include <iostream>
#include <climits> // For INT_MIN and INT_MAX
using namespace std;

// Function prototypes
void enterMarks(int marks[], int N);
double calculateAverage(int marks[], int N);
int findHighest(int marks[], int N);
int findLowest(int marks[], int N);

int main() {
    int N;
    cout << "Enter the number of students: ";
    cin >> N;

    int marks[N]; // Array to store marks of N students
    int choice;

    do {
        // Display menu
        cout << "\nMenu:\n";
        cout << "1. Enter marks of " << N << " students\n";
        cout << "2. Calculate the average marks of the class\n";
        cout << "3. Find the highest and lowest marks\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                enterMarks(marks, N);
                break;
            case 2:
                cout << "Average marks of the class: " << calculateAverage(marks, N) << endl;
                break;
            case 3:
                cout << "Highest marks: " << findHighest(marks, N) << endl;
                cout << "Lowest marks: " << findLowest(marks, N) << endl;
                break;
            case 4:
                cout << "Exiting the program. Goodbye!\n";
                break;
            default:
                cout << "Invalid choice! Please try again.\n";
        }
```



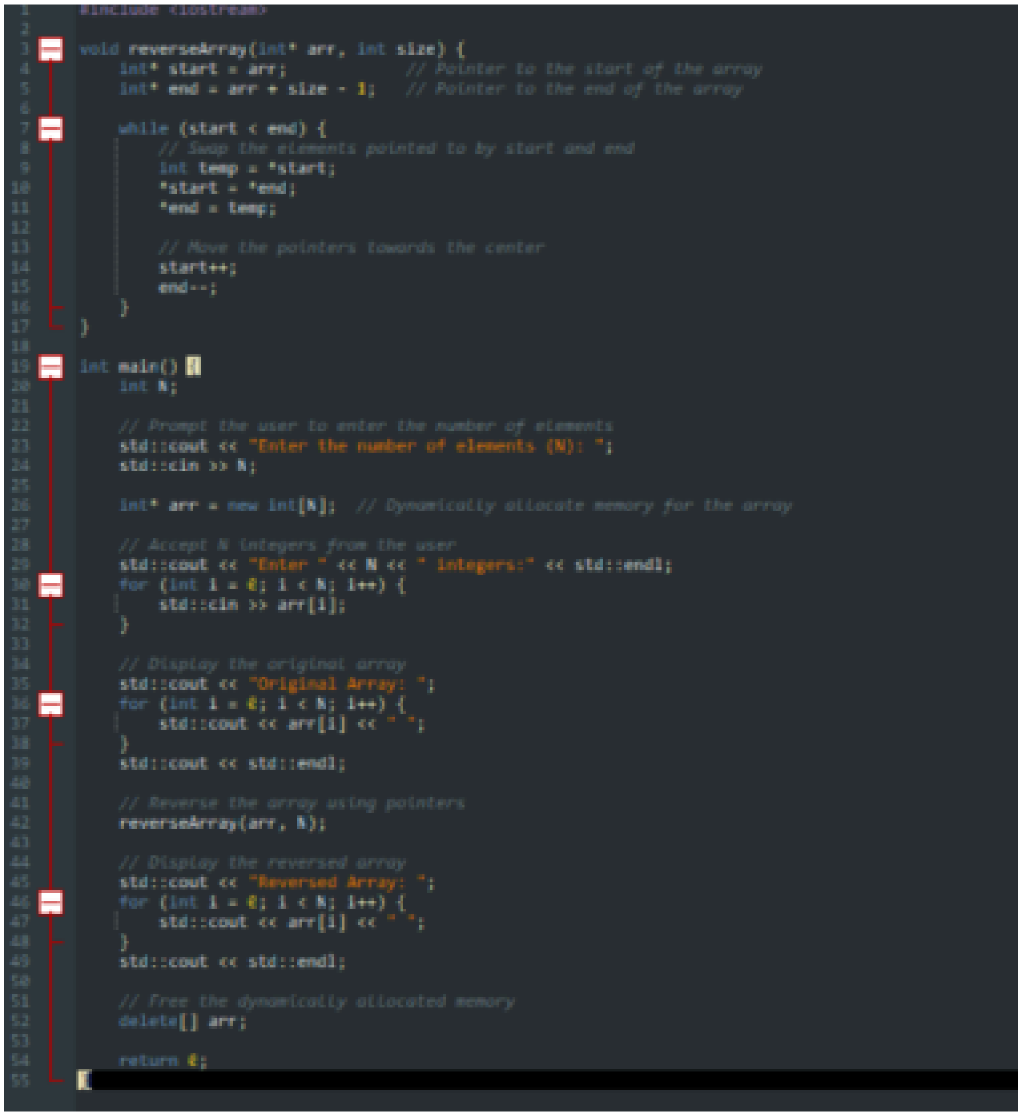
```cpp
        }
    } while (choice != 4);

    return 0;
}

// Function to enter marks of N students
void enterMarks(int marks[], int N) {
    cout << "Enter marks of " << N << " students:\n";
    for (int i = 0; i < N; i++) {
        cout << "Student " << i + 1 << ": ";
        cin >> marks[i];
    }
}

// Function to calculate the average marks of the class
double calculateAverage(int marks[], int N) {
    int sum = 0;
    for (int i = 0; i < N; i++) {
        sum += marks[i];
    }
    return static_cast<double>(sum) / N;
}

// Function to find the highest marks
int findHighest(int marks[], int N) {
    int highest = INT_MIN;
    for (int i = 0; i < N; i++) {
        if (marks[i] > highest) {
            highest = marks[i];
        }
    }
    return highest;
}

// Function to find the lowest marks
int findLowest(int marks[], int N) {
    int lowest = INT_MAX;
    for (int i = 0; i < N; i++) {
        if (marks[i] < lowest) {
            lowest = marks[i];
        }
    }
    return lowest;
}
```

OUTPUT-



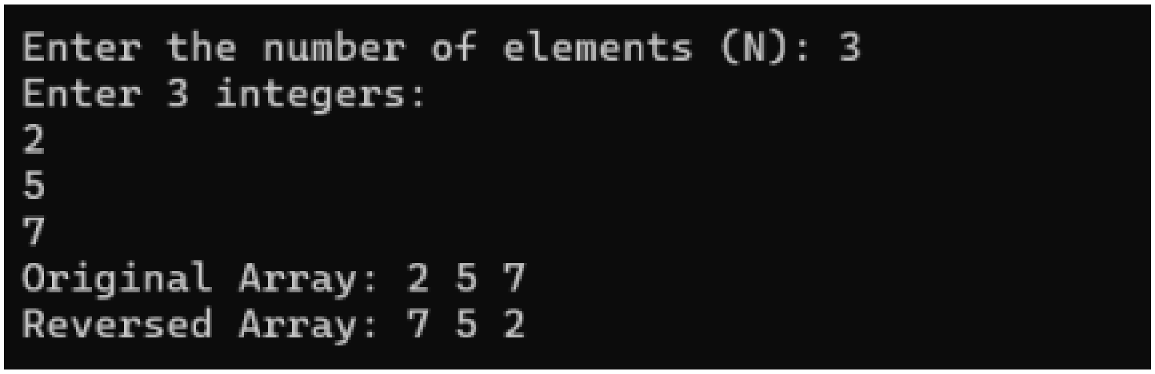
```
Enter the number of students: 10

Menu:
1. Enter marks of 10 students
2. Calculate the average marks of the class
3. Find the highest and lowest marks
4. Exit
Enter your choice: 1
Enter marks of 10 students:
Student 1:
2
Student 2: 5
Student 3: 6
Student 4: 4
Student 5: 5
Student 6: 8
Student 7: 9
Student 8: 10
Student 9: 4
Student 10: 3

Menu:
1. Enter marks of 10 students
2. Calculate the average marks of the class
3. Find the highest and lowest marks
4. Exit
Enter your choice: 3
Highest marks: 10
Lowest marks: 2

Menu:
1. Enter marks of 10 students
2. Calculate the average marks of the class
3. Find the highest and lowest marks
4. Exit
Enter your choice:
```

PROBLEM-3



```cpp
#include <iostream>

void reverseArray(int* arr, int size) {
    int* start = arr;           // Pointer to the start of the array
    int* end = arr + size - 1;  // Pointer to the end of the array

    while (start < end) {
        // Swap the elements pointed to by start and end
        int temp = *start;
        *start = *end;
        *end = temp;

        // Move the pointers towards the center
        start++;
        end--;
    }
}

int main() {
    int N;

    // Prompt the user to enter the number of elements
    std::cout << "Enter the number of elements (N): ";
    std::cin >> N;

    int* arr = new int[N];  // Dynamically allocate memory for the array

    // Accept N integers from the user
    std::cout << "Enter " << N << " integers:" << std::endl;
    for (int i = 0; i < N; i++) {
        std::cin >> arr[i];
    }

    // Display the original array
    std::cout << "Original Array: ";
    for (int i = 0; i < N; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    // Reverse the array using pointers
    reverseArray(arr, N);

    // Display the reversed array
    std::cout << "Reversed Array: ";
    for (int i = 0; i < N; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    // Free the dynamically allocated memory
    delete[] arr;

    return 0;
}
```

OUTPUT-



```
Enter the number of elements (N): 3
Enter 3 integers:
2
5
7
Original Array: 2 5 7
Reversed Array: 7 5 2
```

PROBLEM-4

What is the data type of ‘result’ in the below code? Justify your answer based on C++'s type conversion rules.

```
float x = 2.5;
int y = 3;
auto result = x / y;
```

SOLUTION

In the given code:

The data type of `result` will be `float`. Here's the justification based on C++'s type conversion rules:

1. Operand Types:
   - `x` is of type `float`.
   - `y` is of type `int`.

2. Type Conversion Rules:
   - When performing arithmetic operations between two operands of different types, C++ performs implicit type conversion(also known as usual arithmetic conversions) to convert the operands to a common type.
   - In this case, the division operation (`x / y`) involves a `float` and an `int`. - According to C++ rules, when an `int` is used in an operation with a `float`, the `int` is promoted to a `float` before the operation is performed.

3. Resulting Type-
   - After the promotion, both operands are of type `float`, so the result of the division will also be of type `float`.
   - The `auto` keyword deduces the type of `result` based on the type of the expression `x / y`, which is `float`.

Thus, the type of `result` is `float`.

PROBLEM-5

Consider this code snippet:

```
double pi = 3.14159;
int approx_pi = (int)pi + 0.5;
std::cout << approx_pi;
```

What is the expected output? How would you modify the code to ensure correct rounding to the nearest integer?

SOLUTION

EXPECTED OUTPUT=3

Modified code-

```
#include <iostream>
int main() {
    double pi = 3.14159;
    int approx_pi = (int)(pi + 0.5);
    std::cout << approx_pi;
    return 0;
}
```