ASSESSMENT QUESTION

1.Write a program by using a global variable to track the number of times an element in an array is accessed. The program will introduce additional functionality to handle multiple arrays, element modifications, and statistical analysis of access patterns.

Detailed Requirements:
Global Variable Usage:
Declare a global counter variable, access_count, to track the total number of times elements in one or more arrays are accessed.
Declare an additional global array, access_frequency, to record how many times each element of the main array is accessed.
Array Initialization:
Declare a single-dimensional array of size NNN, where NNN is provided by the user.
Allow the user to initialize the array manually or use a default set of values.
Access Operations:
Implement a menu-driven program that provides the following options:
Read an element:
Prompt the user for an index, retrieve the value at that index, and increment the global counter and the corresponding access_frequency for that index.
Modify an element:
Allow the user to update an element at a specified index. Increment the counters for access and modification.
Access a range of elements:
Allow the user to specify a range of indices, retrieve all elements in that range, and increment the counters for each accessed element.
Statistical Analysis:
After each operation, display the current values of:
The global counter (access_count).
The access_frequency array.
Provide an option to display the most frequently accessed element(s) and the least accessed element(s).
Edge Case Handling:
Ensure that all input indices are within valid bounds.
If an invalid index is accessed or modified, display an error message without incrementing any counters.
Handle cases where the user attempts to access an empty array gracefully.
Reset Functionality:
Allow the user to reset the global counters and frequency array without affecting the main array.
Extended Functionality:
Introduce support for multiple arrays, each with its own access frequency tracking.
Allow switching between arrays to demonstrate the global counter’s shared functionality across all arrays.
Performance Optimization:
Ensure efficient access and modification operations, especially for larger arrays.
Optimize frequency analysis for real-time reporting of most/least accessed elements.
Output Requirements:
After every operation, display:
The updated array (if applicable).
The total access_count.
The access_frequency array.
Provide a summary of access patterns upon exiting the program.

Example Execution:
Input:
Array size N=5N = 5N=5.
Initial array: [10, 20, 30, 40, 50].
Operations:

Read element at index 2.
Modify element at index 3 to 45.
Access elements in the range [1, 4].
Display access statistics.
Processing:
Read Element:
Index 2 accessed.
Increment access_count to 1.
Update access_frequency: [0, 0, 1, 0, 0].
Modify Element:
Index 3 modified to 45.
Increment access_count to 2.
Update access_frequency: [0, 0, 1, 1, 0].
Access Range:
Indices [1, 2, 3, 4] accessed.
Increment access_count to 6.
Update access_frequency: [0, 1, 2, 2, 1].
Output:
Initial Array: [10, 20, 30, 40, 50]

Operation: Read element at index 2
Value: 30
Access Count: 1
Access Frequency: [0, 0, 1, 0, 0]

Operation: Modify element at index 3 to 45
Access Count: 2
Access Frequency: [0, 0, 1, 1, 0]

Operation: Access range [1-4]
Access Count: 6
Access Frequency: [0, 1, 2, 2, 1]

Access Statistics:
Most Accessed Element(s): 30 and 45 (Accessed 2 times)
Least Accessed Element(s): 10 (Accessed 0 times)

ANSWER

CODE:

```
#include <stdio.h>
#include <limits.h>

#define MAX_ARRAYS 10          // Maximum number of arrays the user can create
#define MAX_ARRAY_SIZE 100     // Maximum size of each array

// Global variables to track access counts and frequency
int access_count = 0;            // Total number of accesses across all arrays
int access_frequency[MAX_ARRAY_SIZE];  // Access frequency for each array element
int num_arrays = 0;              // Number of arrays created
int array_size[MAX_ARRAYS];      // Size of each array
int arrays[MAX_ARRAYS][MAX_ARRAY_SIZE];  // Array of arrays (statically allocated)

// Function to read an element from the current array
```

```
void read_element(int array_index, int index) {
    if (index >= 0 && index < array_size[array_index]) {
        access_count++;
        access_frequency[index]++;
        printf("Value at index %d: %d\n", index, arrays[array_index][index]);
    } else {
        printf("Error: Index out of bounds!\n");
    }
}

// Function to modify an element in the current array
void modify_element(int array_index, int index, int new_value) {
    if (index >= 0 && index < array_size[array_index]) {
        arrays[array_index][index] = new_value;
        access_count++;
        access_frequency[index]++;
        printf("Element at index %d modified to %d\n", index, new_value);
    } else {
        printf("Error: Index out of bounds!\n");
    }
}

// Function to access a range of elements in the current array
void access_range(int array_index, int start, int end) {
    if (start >= 0 && end < array_size[array_index] && start <= end) {
        for (int i = start; i <= end; i++) {
            access_count++;
            access_frequency[i]++;
            printf("Value at index %d: %d\n", i, arrays[array_index][i]);
        }
    } else {
        printf("Error: Invalid index range!\n");
    }
}

// Function to display access statistics
void display_statistics(int array_index) {
    printf("\nAccess Count: %d\n", access_count);
    printf("Access Frequency: ");
    for (int i = 0; i < array_size[array_index]; i++) {
        printf("%d ", access_frequency[i]);
    }
    printf("\n");

    // Find most and least accessed elements
    int max_access = INT_MIN, min_access = INT_MAX;
    for (int i = 0; i < array_size[array_index]; i++) {
        if (access_frequency[i] > max_access) max_access = access_frequency[i];
        if (access_frequency[i] < min_access) min_access = access_frequency[i];
    }

    printf("Most Accessed Element(s): ");
    for (int i = 0; i < array_size[array_index]; i++) {
        if (access_frequency[i] == max_access) {
            printf("%d ", arrays[array_index][i]);
```

```
        }
    }

    printf("\nLeast Accessed Element(s): ");
    for (int i = 0; i < array_size[array_index]; i++) {
        if (access_frequency[i] == min_access) {
            printf("%d ", arrays[array_index][i]);
        }
    }

    printf("\n");
}

// Function to reset the global counters and frequency array
void reset_counters() {
    access_count = 0;
    for (int i = 0; i < MAX_ARRAY_SIZE; i++) {
        access_frequency[i] = 0;
    }
    printf("Counters and frequency reset.\n");
}

// Function to switch between multiple arrays
void switch_array() {
    int choice;
    printf("Enter the index of the array to switch to (0 to %d): ", num_arrays - 1);
    scanf("%d", &choice);
    if (choice >= 0 && choice < num_arrays) {
        printf("Switched to array %d\n", choice);
    } else {
        printf("Error: Invalid array index.\n");
    }
}

// Function to create a new array
void create_array() {
    if (num_arrays < MAX_ARRAYS) {
        int size;
        printf("Enter the size of the array: ");
        scanf("%d", &size);

        if (size > MAX_ARRAY_SIZE) {
            printf("Error: Maximum allowed size is %d.\n", MAX_ARRAY_SIZE);
            return;
        }

        array_size[num_arrays] = size;

        printf("Initialize the array with values:\n");
        for (int i = 0; i < size; i++) {
            printf("Enter value for element %d: ", i);
            scanf("%d", &arrays[num_arrays][i]);
        }

        num_arrays++;
```

```
        printf("Array %d created successfully.\n", num_arrays - 1);
    } else {
        printf("Error: Maximum number of arrays reached.\n");
    }
}

// Main program loop
int main() {
    int choice, array_index, index, start, end, new_value;

    while (1) {
        printf("\nMenu:\n");
        printf("1. Create a new array\n");
        printf("2. Switch array\n");
        printf("3. Read an element\n");
        printf("4. Modify an element\n");
        printf("5. Access range of elements\n");
        printf("6. Display access statistics\n");
        printf("7. Reset counters\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                create_array();
                break;
            case 2:
                switch_array();
                break;
            case 3:
                printf("Enter array index: ");
                scanf("%d", &array_index);
                printf("Enter index to read: ");
                scanf("%d", &index);
                read_element(array_index, index);
                display_statistics(array_index);
                break;
            case 4:
                printf("Enter array index: ");
                scanf("%d", &array_index);
                printf("Enter index to modify: ");
                scanf("%d", &index);
                printf("Enter new value: ");
                scanf("%d", &new_value);
                modify_element(array_index, index, new_value);
                display_statistics(array_index);
                break;
            case 5:
                printf("Enter array index: ");
                scanf("%d", &array_index);
                printf("Enter start index: ");
                scanf("%d", &start);
                printf("Enter end index: ");
                scanf("%d", &end);
```

```
            access_range(array_index, start, end);
            display_statistics(array_index);
            break;
        case 6:
            printf("Enter array index: ");
            scanf("%d", &array_index);
            display_statistics(array_index);
            break;
        case 7:
            reset_counters();
            break;
        case 8:
            printf("Exiting program.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  }

  return 0;
}
```