Write a C program to demonstrate the difference in scope between global and local arrays with the same name. Extend the problem with additional requirements to manipulate and analyze the arrays, showcasing how local variables shadow global variables within a specific scope.

Detailed Requirements:

Global and Local Array Declaration:

Declare a global array named data with a size of 10 elements, initialized with static values (e.g., even numbers from 2 to 20).

Declare a local array named data inside a function with a size of 10 elements, initialized by user input.

Array Manipulation:

Global Array:

Calculate the sum of all elements in the global data array and display it.

Multiply each element in the global array by 2 and display the updated array.

Local Array:

Inside the function where the local array data is declared:

Compute the average of the elements.

Replace all elements greater than the average with the value 0.

Display the modified local array.

Scope Demonstration:

Clearly show how the local data array shadows the global data array within the function scope.

Access and display the global data array both before and after entering the function where the local data array is used.

Explicitly access the global array within the local scope

Decision-Making and Looping:

Use loops to process the arrays (e.g., for computing sums, averages, or transformations).

Use decision-making statements (if/else) to identify and modify elements based on the average in the local array.

Edge Case Handling:

For the local array:

If all elements are less than or equal to the average, display "No elements replaced."

If the array contains negative numbers, exclude them from the average calculation.

For the global array:

Ensure the program handles overflow gracefully when multiplying elements.

Output Requirements:

Display the initial global array.

Display the sum and updated global array after manipulation.

Display the local array before and after modifications.

Clearly indicate when the global or local data array is being accessed.

Extended Functionality:

Allow the user to re-run the function with a new local array without reinitializing the global array.

Provide an option to reset the global array to its original state.

Example Execution:

Global Array Initialization:

c

Copy code

data = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

User Input for Local Array:

c

Copy code

Enter 10 integers for the local array: [5, 15, 25, 35, 10, 20, 30, 40, 50, 60]

Processing:

Global Array:

Initial Sum: 2+4+6+8+10+12+14+16+18+20=110

Updated Array (after multiplying by 2): [4, 8, 12, 16, 20, 24, 28, 32, 36, 40]

Local Array:

Average: (5+15+25+35+10+20+30+40+50+60)/10=29

Replacing values greater than 29: [5, 15, 25, 0, 10, 20, 0, 0, 0, 0]

Output:

Global Array (Initial): [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

Global Array Sum: 110

Global Array (Updated): [4, 8, 12, 16, 20, 24, 28, 32, 36, 40]

Local Array (Input): [5, 15, 25, 35, 10, 20, 30, 40, 50, 60]

Local Array Average: 29

Local Array (Modified): [5, 15, 25, 0, 10, 20, 0, 0, 0, 0]

```
#include <stdio.h>

#define ARRAY_SIZE 10
int data[ARRAY_SIZE] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};


int calculateGlobalSum() {
   int sum = 0;
   for (int i = 0; i < ARRAY_SIZE; i++) {
      sum += data[i];
   }
   return sum;
}

void manipulateGlobalArray() {
   printf("Global Array after multiplying by 2:\n");
   for (int i = 0; i < ARRAY_SIZE; i++) {
      data[i] *= 2;
      printf("%d ", data[i]);
   }
   printf("\n");
}

void processLocalArray() {
   int localData[ARRAY_SIZE];
```

```
int sum = 0;
int count = 0;
int total = 0;

printf("Enter 10 integers for the local array:\n");
for (int i = 0; i < ARRAY_SIZE; i++) {
   scanf("%d", &localData[i]);
}
for (int i = 0; i < ARRAY_SIZE; i++) {
   if (localData[i] > 0) {
      sum += localData[i];
      count++;
   }
}
float average = (count > 0) ? (float)sum / count : 0;

printf("Local Array Average: %.2f\n", average);

int replaced = 0;
for (int i = 0; i < ARRAY_SIZE; i++) {
   if (localData[i] > average) {
      localData[i] = 0;
      replaced++;
   }
}


printf("Local Array Modified:\n");
for (int i = 0; i < ARRAY_SIZE; i++) {
   printf("%d ", localData[i]);
}
```

```
    printf("\n");

    if (replaced == 0) {
        printf("No elements replaced in the local array.\n");
    }
}

void resetGlobalArray() {
    int originalData[ARRAY_SIZE] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
    for (int i = 0; i < ARRAY_SIZE; i++) {
        data[i] = originalData[i];
    }
}

int main() {
    int choice;

    while (1) {
        printf("\n=== Menu ===\n");
        printf("1. Display Global Array\n");
        printf("2. Process Local Array\n");
        printf("3. Manipulate Global Array\n");
        printf("4. Reset Global Array\n");
        printf("5. Exit\n");
        printf("Choose an option: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:

                printf("Global Array :\n");
```

```
            for (int i = 0; i < ARRAY_SIZE; i++) {
                printf("%d ", data[i]);
            }
            printf("\n");
            printf("Global Array Sum: %d\n", calculateGlobalSum());
            break;

        case 2:

            processLocalArray();
            break;

        case 3:

            manipulateGlobalArray();
            break;

        case 4:

            resetGlobalArray();
            printf("Global array reset to its original values.\n");
            break;

        case 5:

            printf("Exiting the program.\n");
            return 0;

        default:
            printf("Invalid choice. Please try again.\n");
    }
```

```
    }

    return 0;
}
```