Exam 1: 30/12/2024

Q **Objective**:
Write a C program to process a one-dimensional array of integers. For each element in the array, check if it exceeds a specified **threshold value**. If it does, replace the element with **0**. Introduce additional complexity by incorporating multiple conditions and dynamic thresholds.

Detailed Requirements:

1. **Input Constraints**:
   - The program should accept an integer array from the user, where the size (N) is specified at runtime.
   - The size of the array must be between **10 and 200**, inclusive. Validate the size and prompt the user until a valid input is provided.
   - All array elements must be integers, including negative numbers.
2. **Threshold Definition**:
   - The user must provide a **global threshold value** (T) between 10 and 1000, inclusive. If the input threshold is invalid, prompt the user until it is valid.
   - The program should also compute a **dynamic threshold** for each element, defined as Dynamic Threshold=T×Index of the Element\text{Dynamic Threshold} = \text{T} \times \text{Index of the Element}Dynamic Threshold=T×Index of the Element. For example, if T=50T = 50T=50, the dynamic threshold for the 3rd element (index 2) will be 50×2=10050 \times 2 = 10050×2=100.
3. **Replacement Conditions**:
   - If an element exceeds the **global threshold (T)**, replace it with **0**.
   - If an element exceeds its **dynamic threshold**, replace it with **-1**.
   - If an element satisfies both conditions (i.e., exceeds both thresholds), prioritize replacing it with **-1**.
4. **Output Requirements**:
   - Display the original array.
   - Display the modified array after processing.
   - Highlight the indices and original values of elements that were replaced along with the reason (e.g., exceeded global or dynamic threshold).
5. **Edge Case Handling**:
   - If no elements exceed any threshold, the program should display: "No elements exceeded the given thresholds."
   - Negative numbers in the array should also be checked against the thresholds.
6. **Performance Considerations**:
   - Ensure the program efficiently handles arrays close to the maximum size of 200 elements.
   - Avoid unnecessary recomputations of the dynamic threshold by precomputing and storing it for each index.
7. **Additional Constraints**:
   - Use a single loop to traverse the array and perform the necessary replacements.

- Implement separate functions to validate inputs, compute dynamic thresholds, and process the array.

Example Execution:

Input 1:

- **Array Size**: 6
- **Array Elements**: [150, 50, 90, 200, 30, -5]
- **Threshold (T)**: 50

Processing:

- Dynamic Thresholds: [0, 50, 100, 150, 200, 250]
- Replacement Logic:
  - 150>50150 > 50150>50: Replace with 0 (Global Threshold Exceeded)
  - 50=5050 = 5050=50: No Replacement
  - 90<10090 < 10090<100: No Replacement
  - 200>150200 > 150200>150: Replace with -1 (Dynamic Threshold Exceeded)
  - 30<20030 < 20030<200: No Replacement
  - −5<0-5 < 0−5<0: No Replacement

Output:

- **Original Array**: [150, 50, 90, 200, 30, -5]
- **Modified Array**: [0, 50, 90, -1, 30, -5]
- **Replacements**:
  - Index 0: Value 150 replaced with 0 (Global Threshold Exceeded).
-

Program:

```
#include <stdio.h>

int threshold;


int main() {

   int n;


   // Input size of the array
```

```
printf("Enter size of array (10-200):\n");
scanf("%d", &n);
printf("Enter Threshold between(10 and 1000)\n");
scanf("%d",&threshold);
if(threshold<10 || threshold>1000){
   printf("Enter a valid threshold range");
   return 1;
}
// Validate array size
if (n >= 10 && n <= 200) {
   int arr[n];
   int modified_arr[n];
   int dynamic_threshold[n];

   // Input array elements
   printf("Enter %d elements for the array:\n", n);
   for (int i = 0; i < n; i++) {
      printf("arr[%d]: ", i + 1);
      scanf("%d", &arr[i]);
   }

   // Compute dynamic thresholds
   for (int i = 0; i < n; i++) {
      dynamic_threshold[i] = threshold * i;
   }

   // Print dynamic thresholds
   printf("Dynamic thresholds:\n");
   for (int i = 0; i < n; i++) {
```

```
    printf("%d ", dynamic_threshold[i]);
}
printf("\n");

// Process the array
for (int i = 0; i < n; i++) {
    if (arr[i] > threshold && arr[i] > dynamic_threshold[i]) {
        modified_arr[i] = -1; // Both thresholds exceeded
    } else if (arr[i] > dynamic_threshold[i]) {
        modified_arr[i] = -1; // Dynamic threshold exceeded
    } else if (arr[i] > threshold) {
        modified_arr[i] = 0; // Global threshold exceeded
    } else {
        modified_arr[i] = arr[i]; // No replacement
    }
}

// Print the original array
printf("Original Array:\n");
for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
printf("\n");

// Print the modified array
printf("Modified Array:\n");
for (int i = 0; i < n; i++) {
    printf("%d ", modified_arr[i]);
}
```

```
        printf("\n");
    } else {
        printf("Invalid size. Please enter a size between 10 and 200.\n");
    }

    return 0;
}
```