ALGORITHMS LABORATORY
[CS39001]
Individual Work

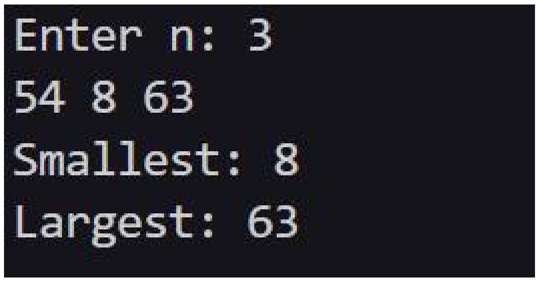
Lab. No.- 2 **Date. : 21.07.2025**

Roll Number:	**2328159**	**Branch/Section:**	**CSSE-1**
Name in Capital:	**AYUSH RAJ**		

Program No: 1.1

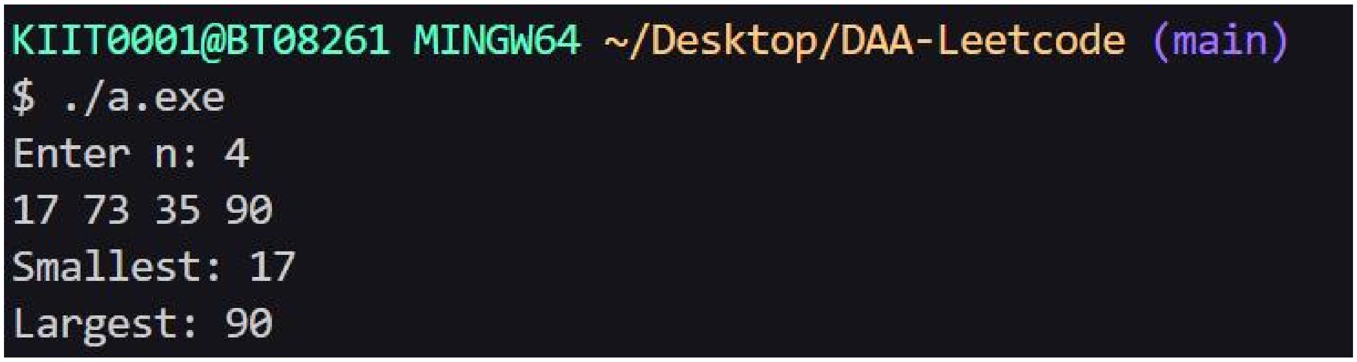
Program Title:
Write a program to store random numbers into an array of n integers and then find out the smallest and largest number stored in it. n is the user input.

Input/Output Screenshots:
RUN-1:



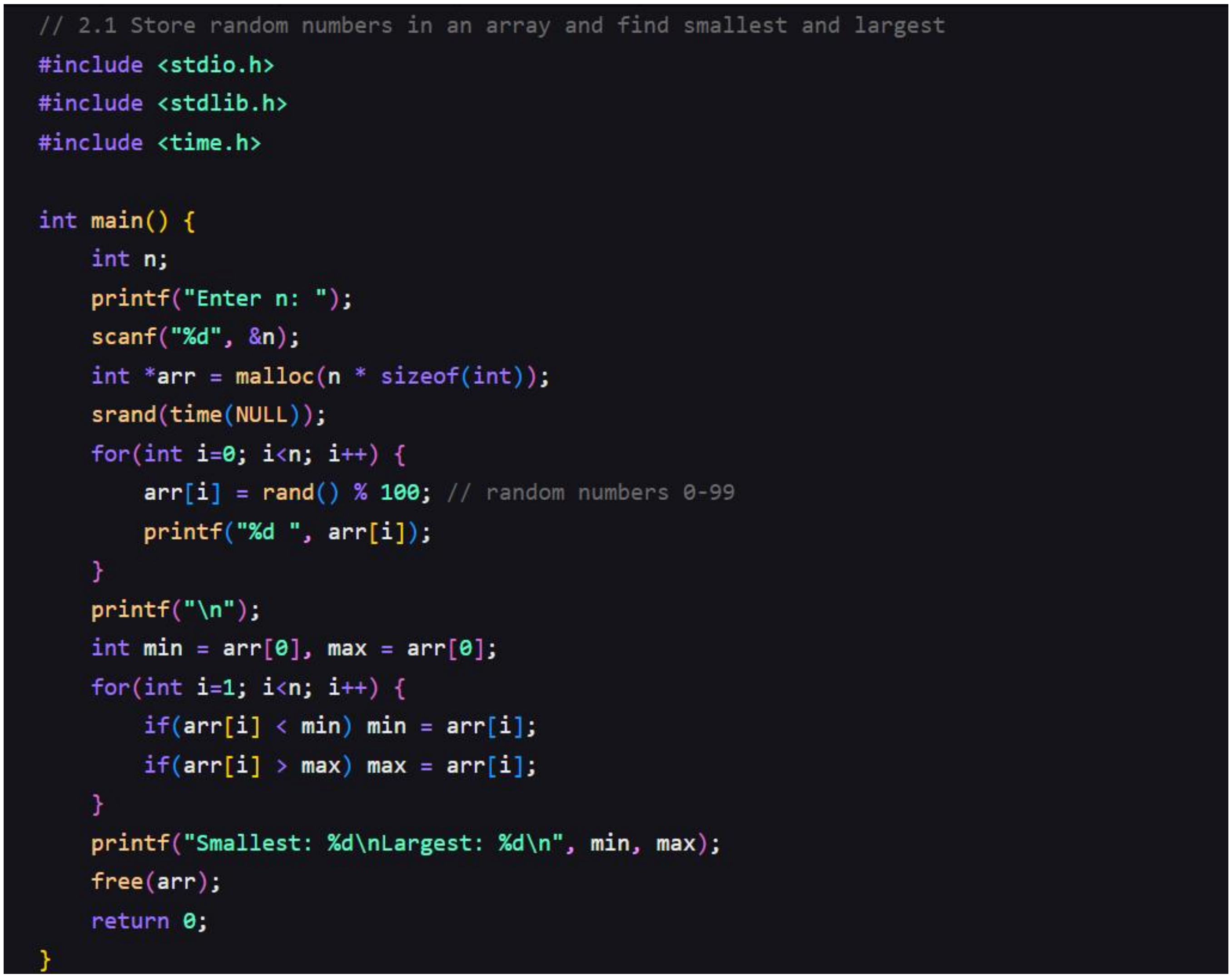
```
Enter n: 3
54 8 63
Smallest: 8
Largest: 63
```

RUN-2



```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode (main)
$ ./a.exe
Enter n: 4
17 73 35 90
Smallest: 17
Largest: 90
```

Source code



```c
// 2.1 Store random numbers in an array and find smallest and largest
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    int *arr = malloc(n * sizeof(int));
    srand(time(NULL));
    for(int i=0; i<n; i++) {
        arr[i] = rand() % 100; // random numbers 0-99
        printf("%d ", arr[i]);
    }
    printf("\n");
    int min = arr[0], max = arr[0];
    for(int i=1; i<n; i++) {
        if(arr[i] < min) min = arr[i];
        if(arr[i] > max) max = arr[i];
    }
    printf("Smallest: %d\nLargest: %d\n", min, max);
    free(arr);
    return 0;
}
```

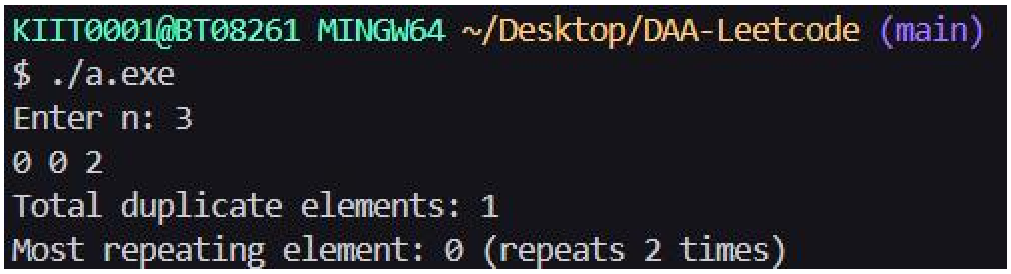
Conclusion/Observation

This program successfully demonstrates how to store random numbers in an array of user-defined size, and efficiently finds both the smallest and largest numbers in the array. It uses dynamic memory allocation for flexibility and ensures proper resource management by freeing the allocated memory at the end. The approach is simple, efficient, and suitable for basic array operations in C.

Program No: 1.2

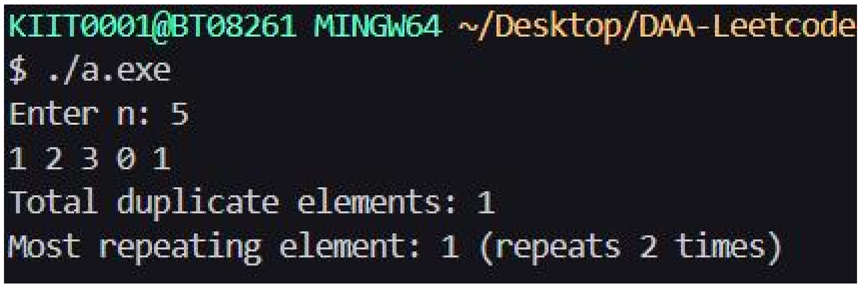
Program Title:
Write a program to store random numbers into an array of n integers, where the array must contains some duplicates. Do the following:
a) Find out the total number of duplicate elements.
b) Find out the most repeating element in the array.

Input/Output Screenshots:
RUN-1:



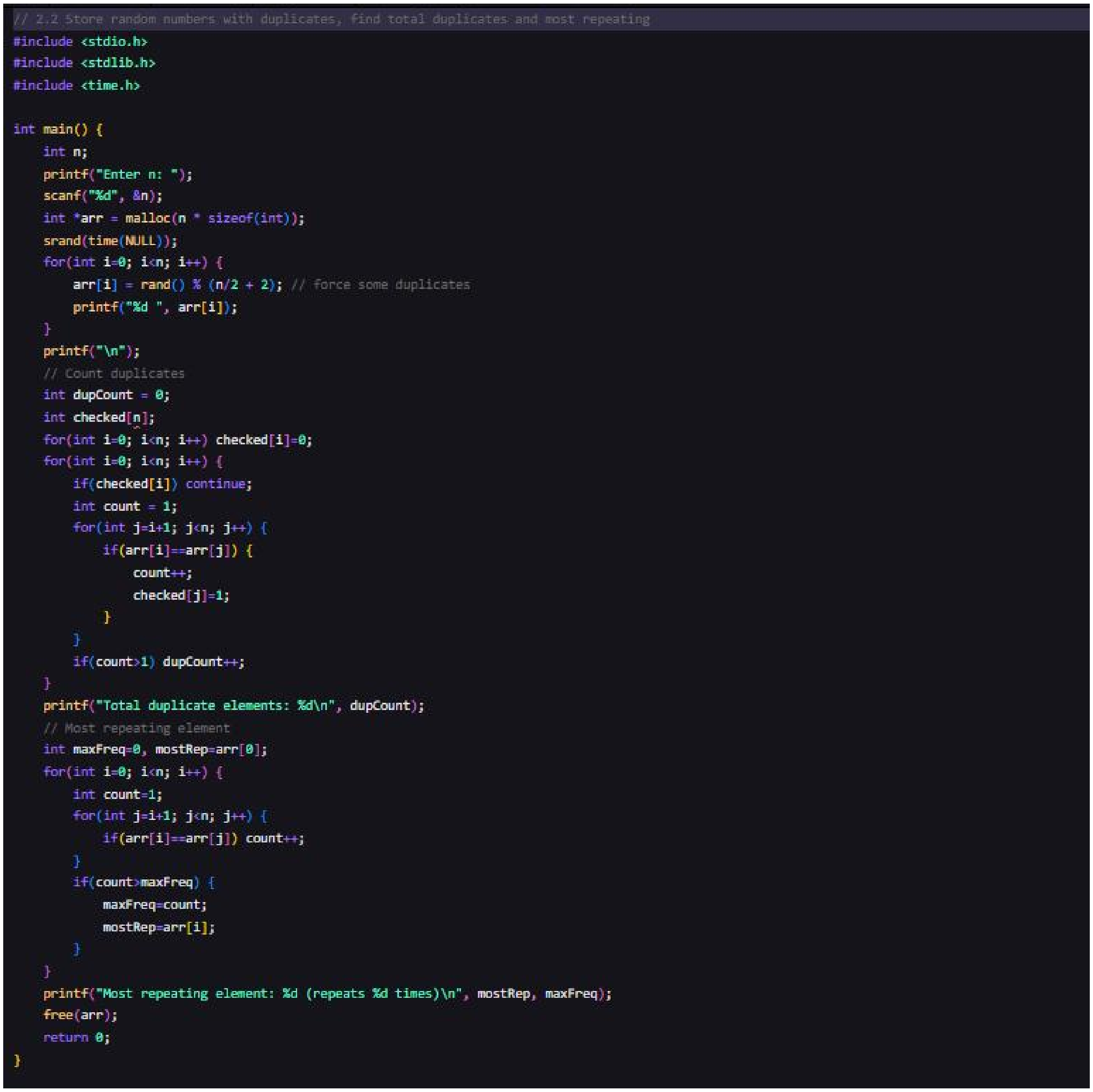
```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode (main)
$ ./a.exe
Enter n: 3
0 0 2
Total duplicate elements: 1
Most repeating element: 0 (repeats 2 times)
```

RUN-2



```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode
$ ./a.exe
Enter n: 5
1 2 3 0 1
Total duplicate elements: 1
Most repeating element: 1 (repeats 2 times)
```

Source code



```c
// 2.2 Store random numbers with duplicates, find total duplicates and most repeating
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    int *arr = malloc(n * sizeof(int));
    srand(time(NULL));
    for(int i=0; i<n; i++) {
        arr[i] = rand() % (n/2 + 2); // force some duplicates
        printf("%d ", arr[i]);
    }
    printf("\n");
    // Count duplicates
    int dupCount = 0;
    int checked[n];
    for(int i=0; i<n; i++) checked[i]=0;
    for(int i=0; i<n; i++) {
        if(checked[i]) continue;
        int count = 1;
        for(int j=i+1; j<n; j++) {
            if(arr[i]==arr[j]) {
                count++;
                checked[j]=1;
            }
        }
        if(count>1) dupCount++;
    }
    printf("Total duplicate elements: %d\n", dupCount);
    // Most repeating element
    int maxFreq=0, mostRep=arr[0];
    for(int i=0; i<n; i++) {
        int count=1;
        for(int j=i+1; j<n; j++) {
            if(arr[i]==arr[j]) count++;
        }
        if(count>maxFreq) {
            maxFreq=count;
            mostRep=arr[i];
        }
    }
    printf("Most repeating element: %d (repeats %d times)\n", mostRep, maxFreq);
    free(arr);
    return 0;
}
```

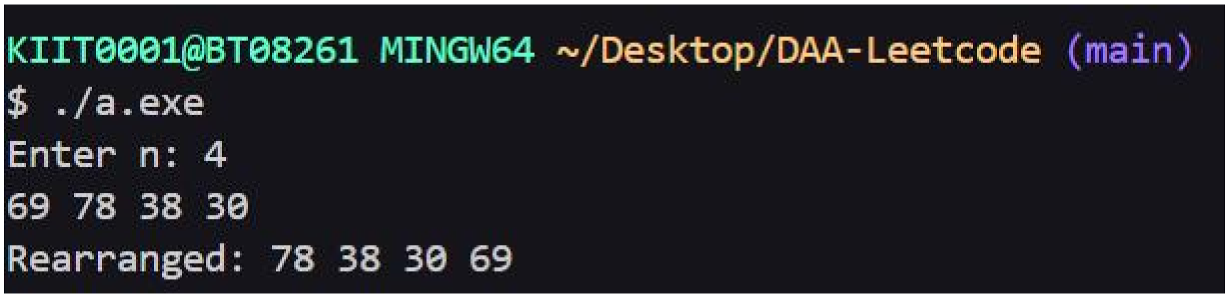
Conclusion/Observation

This program generates an array of random integers with intentional duplicates, then efficiently counts the total number of duplicate elements and identifies the most frequently occurring value. It demonstrates basic frequency analysis and duplicate detection in arrays using C.

Program No: 1.3

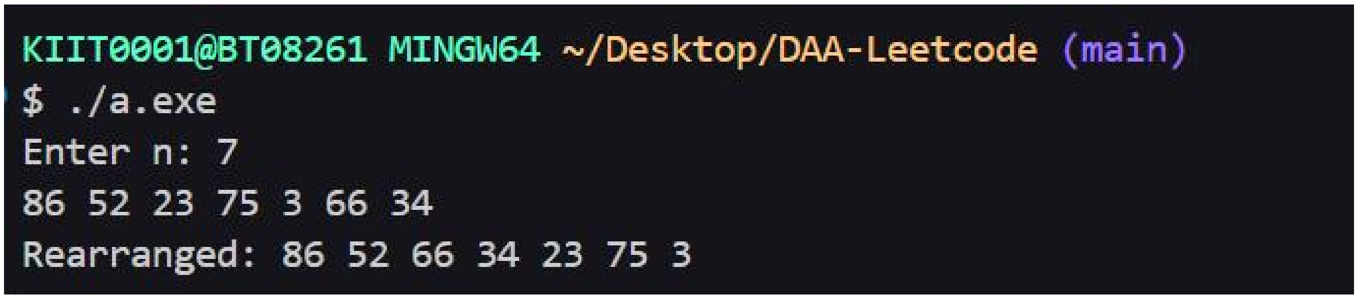
Program Title:
Write a program to rearrange the elements of an array of n integers (random input) such that all even numbers are followed by all odd numbers.

Input/Output Screenshots:
RUN-1:



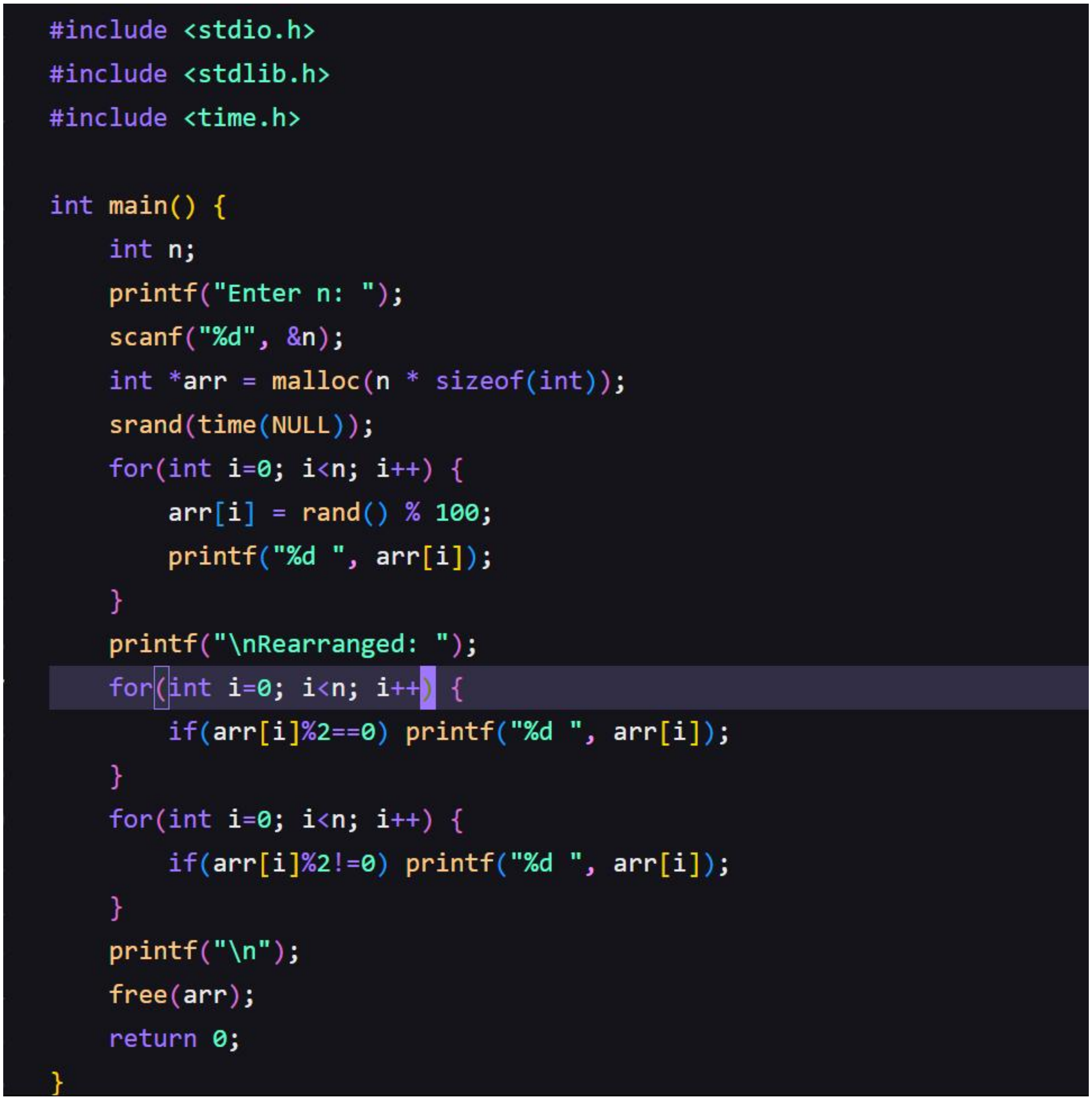
```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode (main)
$ ./a.exe
Enter n: 4
69 78 38 30
Rearranged: 78 38 30 69
```

RUN-2



```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode (main)
$ ./a.exe
Enter n: 7
86 52 23 75 3 66 34
Rearranged: 86 52 66 34 23 75 3
```

Source code



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    int *arr = malloc(n * sizeof(int));
    srand(time(NULL));
    for(int i=0; i<n; i++) {
        arr[i] = rand() % 100;
        printf("%d ", arr[i]);
    }
    printf("\nRearranged: ");
    for(int i=0; i<n; i++) {
        if(arr[i]%2==0) printf("%d ", arr[i]);
    }
    for(int i=0; i<n; i++) {
        if(arr[i]%2!=0) printf("%d ", arr[i]);
    }
    printf("\n");
    free(arr);
    return 0;
}
```

Conclusion/Observation

This program rearranges the elements of a randomly generated integer array so that all even numbers appear before all odd numbers. It showcases simple array traversal and conditional logic to achieve the required ordering without using extra space.

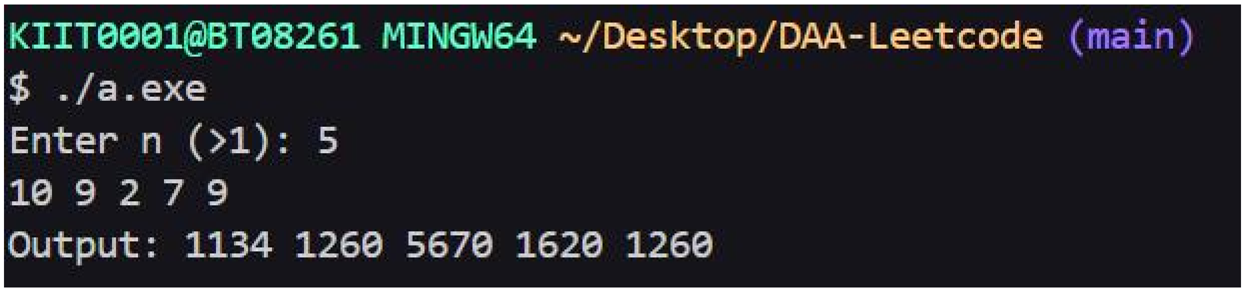
Program No: 1.4

Program Title:

Write a program to display an array of n integers (random input) (n>1), where at every index of the array should contain the product of all elements in the array except the element at the given index. No additional array declaration is allowed.
Example:
Array input : 10, 4, 1, 6, 2
Array output : 48,120,480,80,240

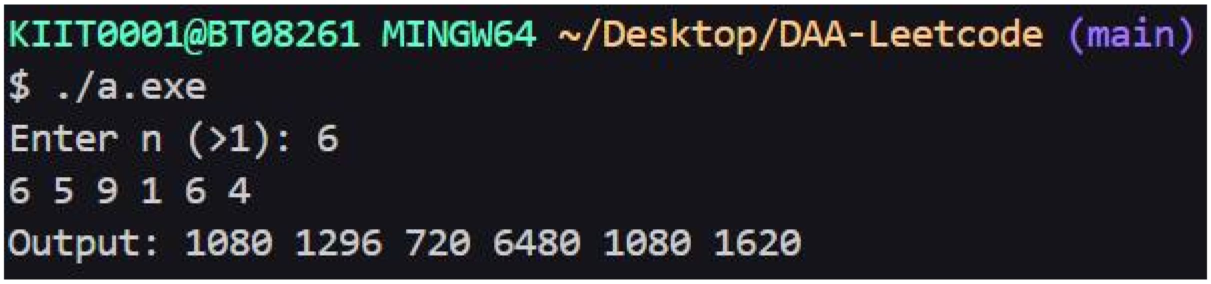
Input/Output Screenshots:

RUN-1:



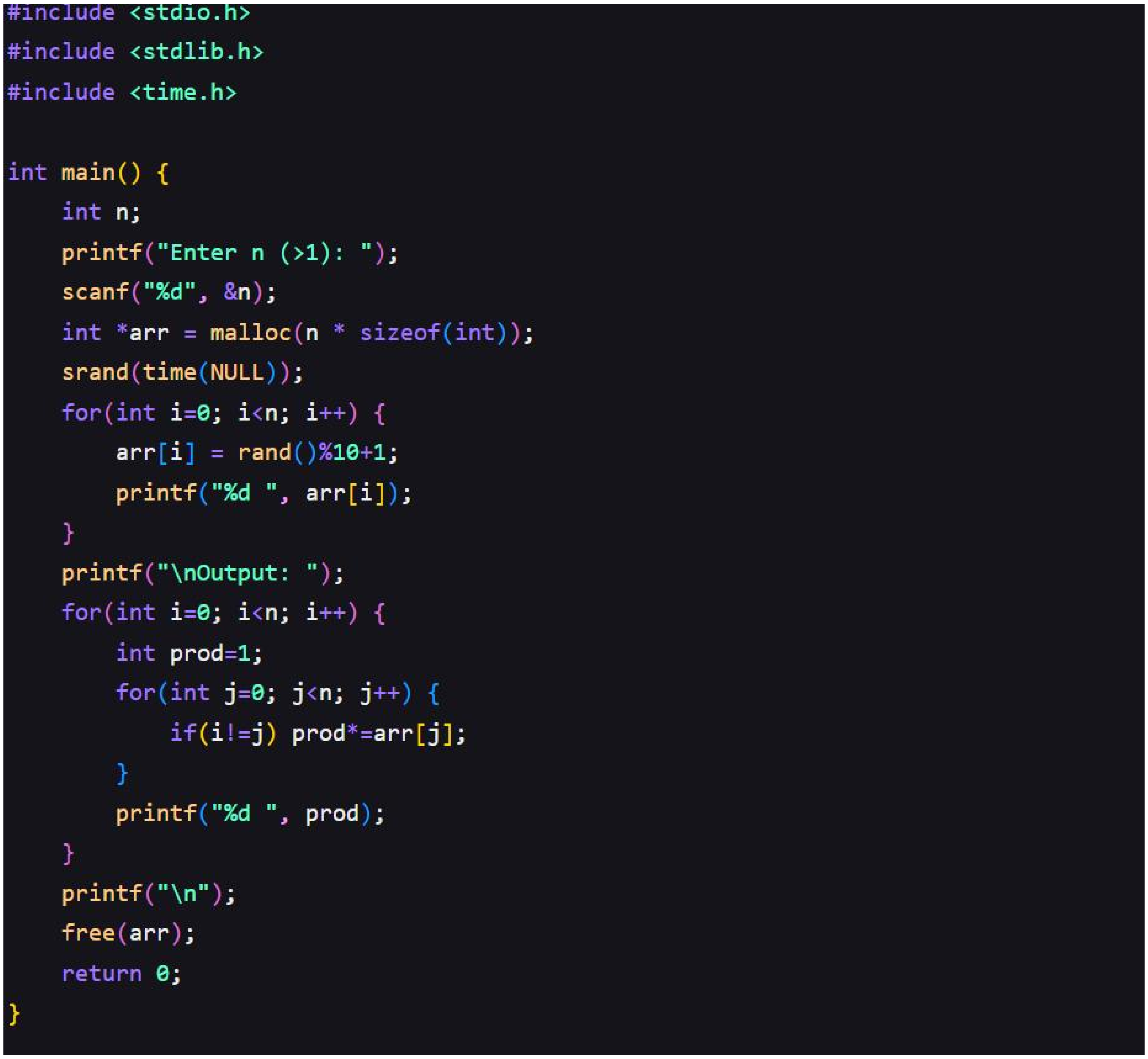
```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode (main)
$ ./a.exe
Enter n (>1): 5
10 9 2 7 9
Output: 1134 1260 5670 1620 1260
```

RUN-2



```
KIIT0001@BT08261 MINGW64 ~/Desktop/DAA-Leetcode (main)
$ ./a.exe
Enter n (>1): 6
6 5 9 1 6 4
Output: 1080 1296 720 6480 1080 1620
```

Source code



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int n;
    printf("Enter n (>1): ");
    scanf("%d", &n);
    int *arr = malloc(n * sizeof(int));
    srand(time(NULL));
    for(int i=0; i<n; i++) {
        arr[i] = rand()%10+1;
        printf("%d ", arr[i]);
    }
    printf("\nOutput: ");
    for(int i=0; i<n; i++) {
        int prod=1;
        for(int j=0; j<n; j++) {
            if(i!=j) prod*=arr[j];
        }
        printf("%d ", prod);
    }
    printf("\n");
    free(arr);
    return 0;
}
```

Conclusion/Observation

This program displays an array where each index contains the product of all elements except the one at that index, using only the original array and no additional arrays. It demonstrates nested loops and careful index management to solve a classic array manipulation problem in C.