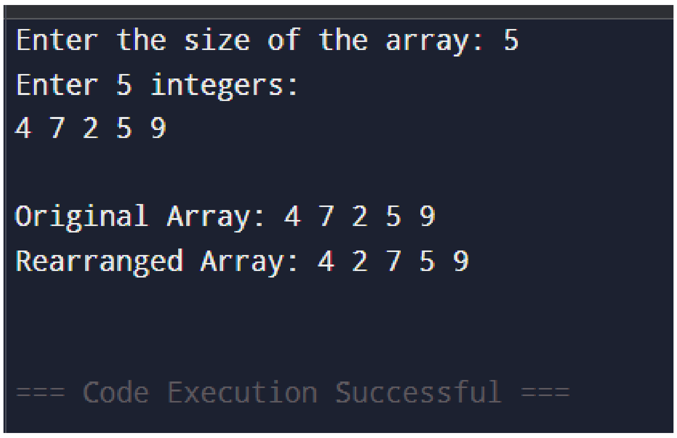
Topic 1: Structures
C program that represents a calendar for a week. Each day has: dayName (e.g., "Monday") tasks (array of strings with maximum 3 tasks per day)
1. Define appropriate structures.
2. Allow the user to input tasks for any day.
3. Display all tasks grouped by the day.

Output:

```
--- Task Entry Menu ---
Choose a day to add tasks:
1. Monday
2. Tuesday
3. Wednesday
4. Thursday
5. Friday
6. Saturday
7. Sunday
8. Display All Tasks
9. Exit
Enter your choice: 1
Enter task for Monday: Go to college
Task added.

--- Task Entry Menu ---
Choose a day to add tasks:
1. Monday
2. Tuesday
3. Wednesday
4. Thursday
5. Friday
6. Saturday
7. Sunday
8. Display All Tasks
9. Exit
Enter your choice: 2
Enter task for Tuesday: watch movie and sleep
Task added.

--- Task Entry Menu ---
Choose a day to add tasks:
1. Monday
2. Tuesday
3. Wednesday
4. Thursday
5. Friday
6. Saturday
7. Sunday
8. Display All Tasks
9. Exit
Enter your choice: 3
Enter task for Wednesday: Eat new food
Task added.

--- Task Entry Menu ---
```

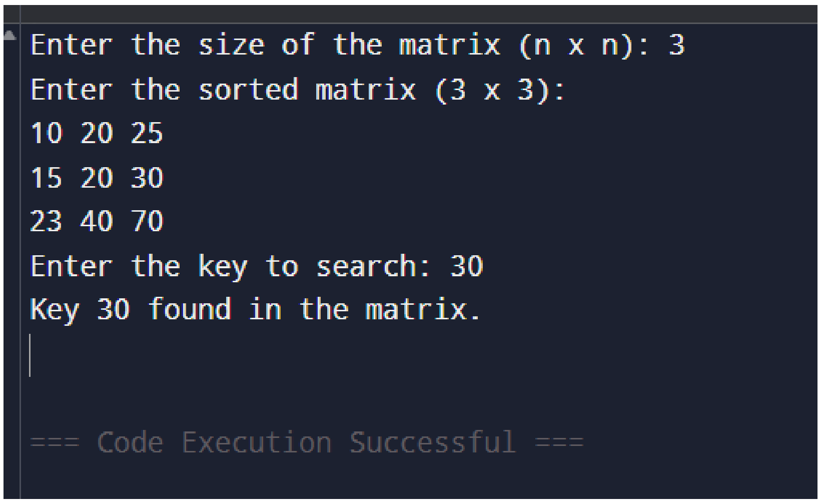
```
Choose a day to add tasks:
1. Monday
2. Tuesday
3. Wednesday
4. Thursday
5. Friday
6. Saturday
7. Sunday
8. Display All Tasks
9. Exit
Enter your choice: 8

--- Weekly Tasks ---

Monday:
  - Go to college

Tuesday:
  - watch movie and sleep

Wednesday:
  - Eat new food

Thursday:
  No tasks.

Friday:
  No tasks.

Saturday:
  No tasks.

Sunday:
  No tasks.

--- Task Entry Menu ---
Choose a day to add tasks:
1. Monday
2. Tuesday
3. Wednesday
4. Thursday
5. Friday
6. Saturday
7. Sunday
8. Display All Tasks
9. Exit
Enter your choice: 9
Exiting...
```

Topic 2: Pointers
Write a function in C that takes a pointer to an integer array and its size, and then rearranges the array in-place such that all even numbers appear before odd numbers, preserving the original relative order using only pointer arithmetic (no indexing with []).



```
Enter the size of the array: 5
Enter 5 integers:
4 7 2 5 9

Original Array: 4 7 2 5 9
Rearranged Array: 4 2 7 5 9


=== Code Execution Successful ===
```

Topic 3: Arrays
You are given a 2D matrix of size n x n where each row and each column is sorted in increasing order. Write a C function to determine whether a given key exists in the matrix using the most efficient approach.



```
Enter the size of the matrix (n x n): 3
Enter the sorted matrix (3 x 3):
10 20 25
15 20 30
23 40 70
Enter the key to search: 30
Key 30 found in the matrix.


=== Code Execution Successful ===
```