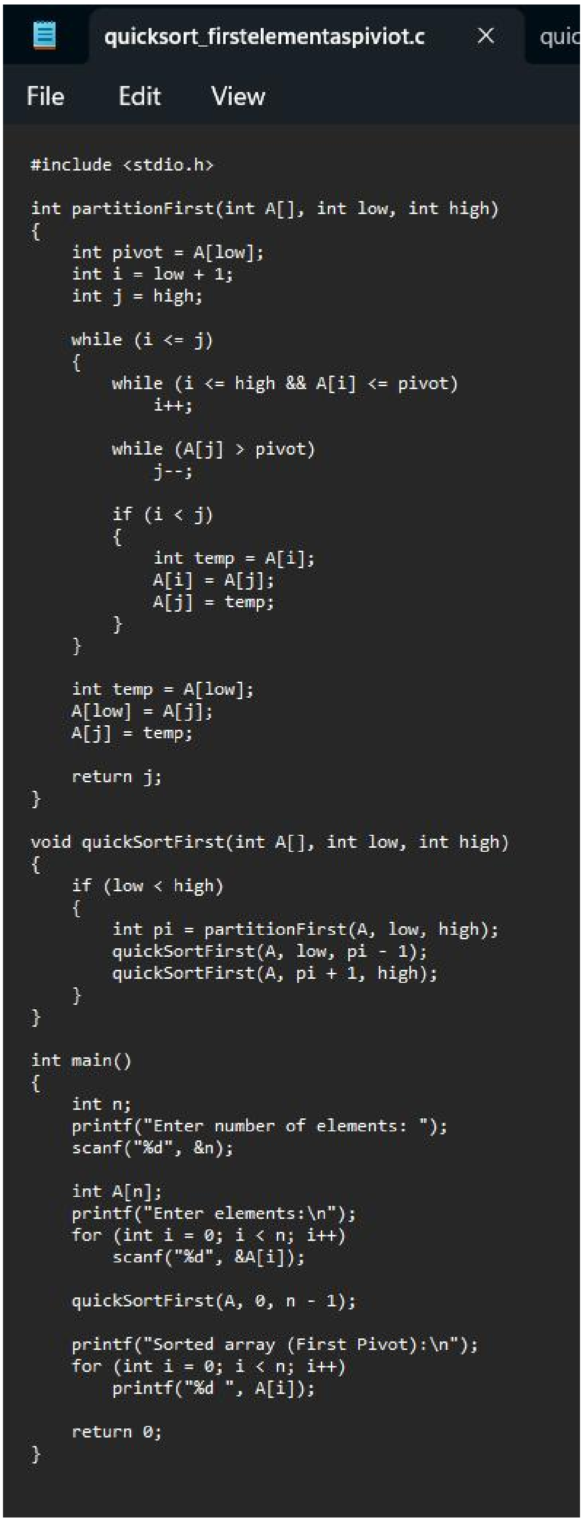
WEEK - 5

Kavin.J.S
CH.SC.U4CSE24119

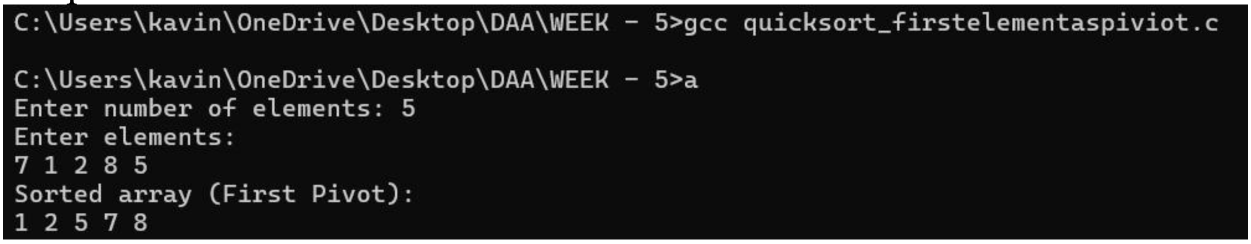
QuickSort : First Element as Piviot:

Code:



```c
#include <stdio.h>

int partitionFirst(int A[], int low, int high)
{
    int pivot = A[low];
    int i = low + 1;
    int j = high;

    while (i <= j)
    {
        while (i <= high && A[i] <= pivot)
            i++;

        while (A[j] > pivot)
            j--;

        if (i < j)
        {
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    }

    int temp = A[low];
    A[low] = A[j];
    A[j] = temp;

    return j;
}

void quickSortFirst(int A[], int low, int high)
{
    if (low < high)
    {
        int pi = partitionFirst(A, low, high);
        quickSortFirst(A, low, pi - 1);
        quickSortFirst(A, pi + 1, high);
    }
}

int main()
{
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    int A[n];
    printf("Enter elements:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &A[i]);

    quickSortFirst(A, 0, n - 1);

    printf("Sorted array (First Pivot):\n");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);

    return 0;
}
```

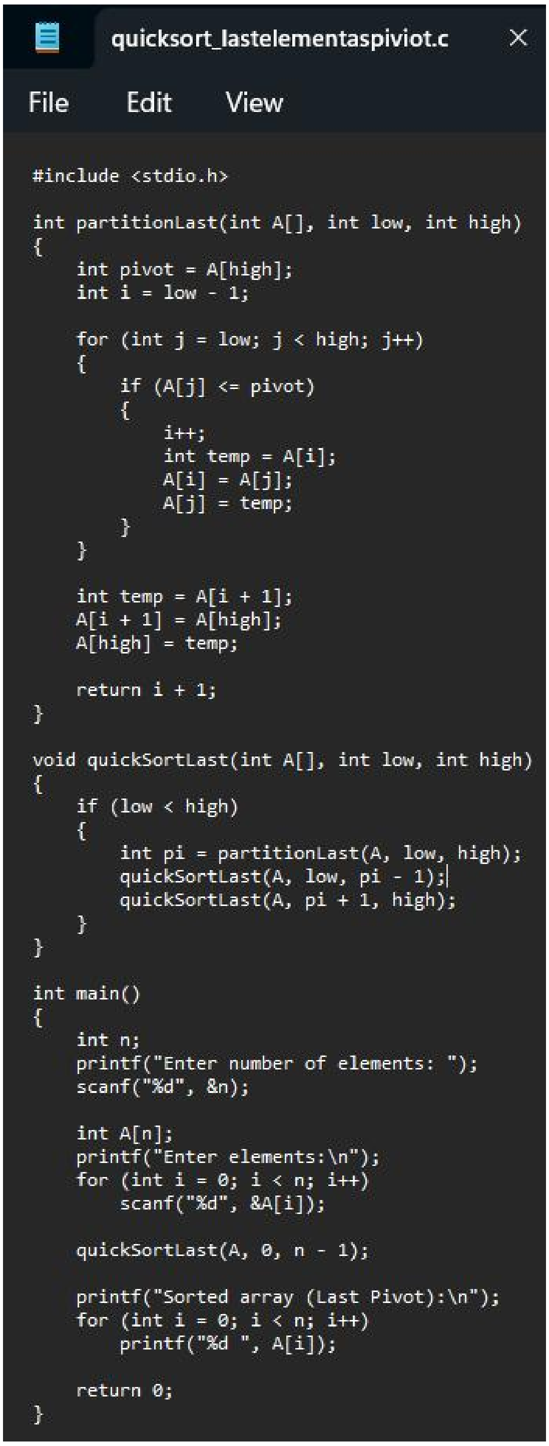
Output:



```
C:\Users\kavin\OneDrive\Desktop\DAA\WEEK - 5>gcc quicksort_firstelementaspiviot.c

C:\Users\kavin\OneDrive\Desktop\DAA\WEEK - 5>a
Enter number of elements: 5
Enter elements:
7 1 2 8 5
Sorted array (First Pivot):
1 2 5 7 8
```

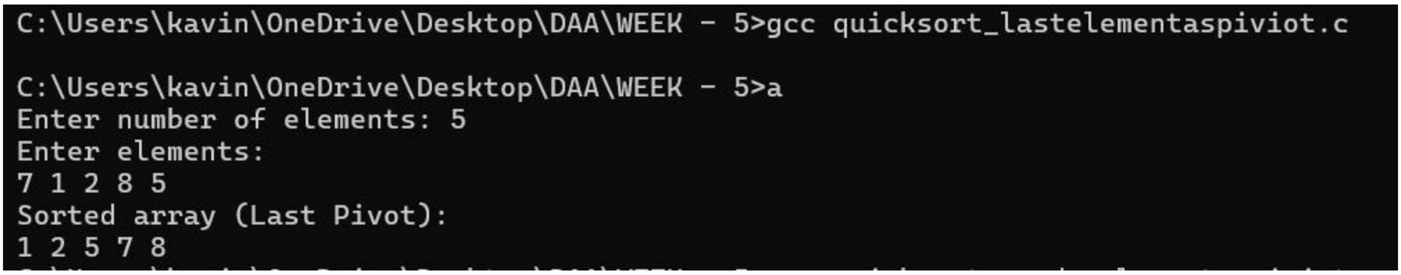
QuickSort : Last Element as Piviot:

Code:



```c
#include <stdio.h>

int partitionLast(int A[], int low, int high)
{
    int pivot = A[high];
    int i = low - 1;

    for (int j = low; j < high; j++)
    {
        if (A[j] <= pivot)
        {
            i++;
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    }

    int temp = A[i + 1];
    A[i + 1] = A[high];
    A[high] = temp;

    return i + 1;
}

void quickSortLast(int A[], int low, int high)
{
    if (low < high)
    {
        int pi = partitionLast(A, low, high);
        quickSortLast(A, low, pi - 1);
        quickSortLast(A, pi + 1, high);
    }
}

int main()
{
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    int A[n];
    printf("Enter elements:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &A[i]);

    quickSortLast(A, 0, n - 1);

    printf("Sorted array (Last Pivot):\n");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);

    return 0;
}
```

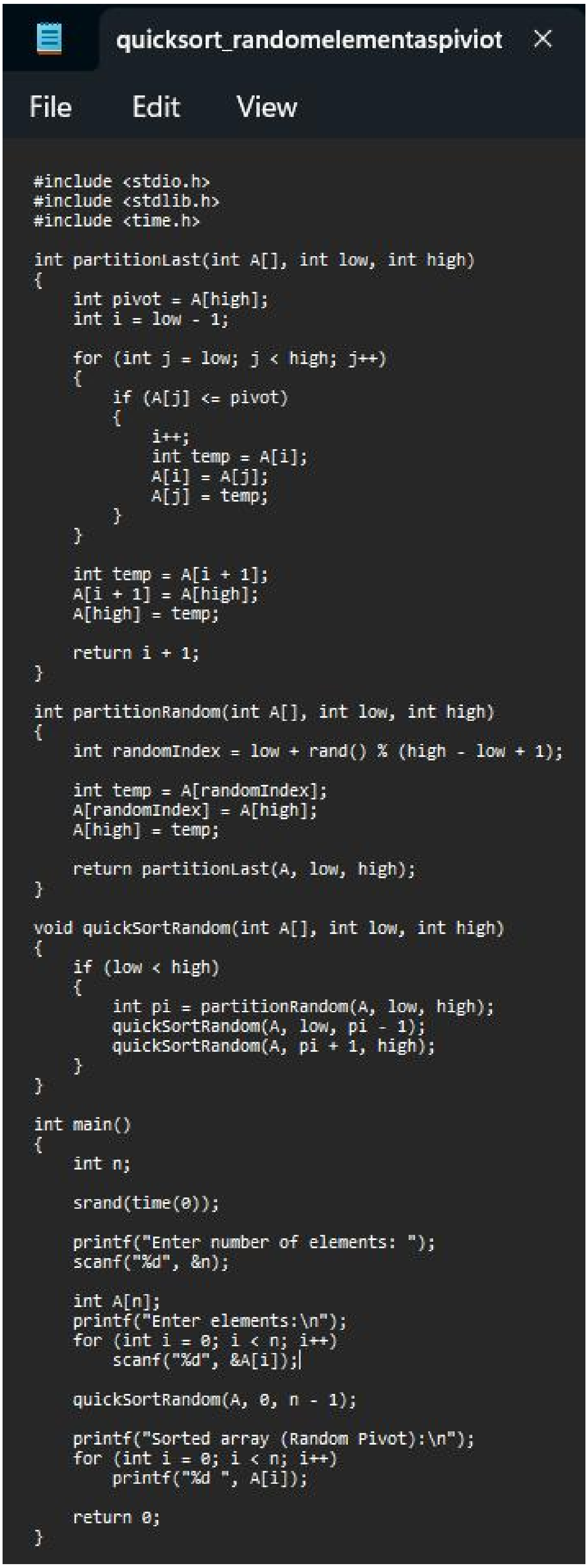
Output:



```
C:\Users\kavin\OneDrive\Desktop\DAA\WEEK - 5>gcc quicksort_lastelementaspiviot.c

C:\Users\kavin\OneDrive\Desktop\DAA\WEEK - 5>a
Enter number of elements: 5
Enter elements:
7 1 2 8 5
Sorted array (Last Pivot):
1 2 5 7 8
```

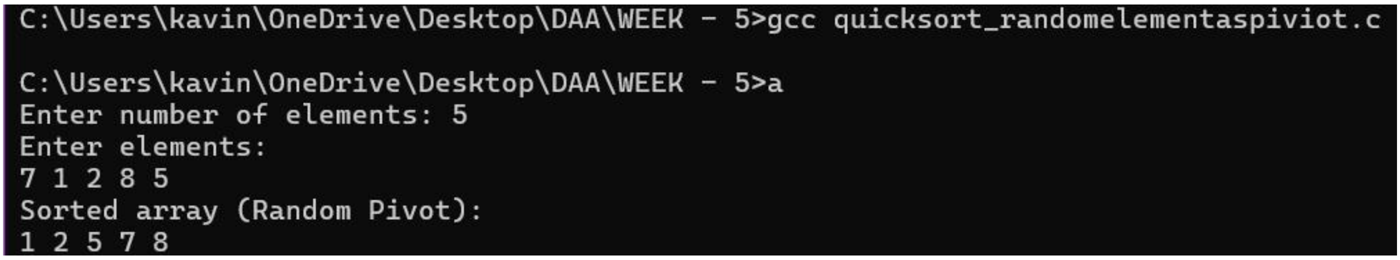
QuickSort : Random Element as Piviot:

Code:



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

int partitionLast(int A[], int low, int high)
{
    int pivot = A[high];
    int i = low - 1;

    for (int j = low; j < high; j++)
    {
        if (A[j] <= pivot)
        {
            i++;
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    }

    int temp = A[i + 1];
    A[i + 1] = A[high];
    A[high] = temp;

    return i + 1;
}

int partitionRandom(int A[], int low, int high)
{
    int randomIndex = low + rand() % (high - low + 1);

    int temp = A[randomIndex];
    A[randomIndex] = A[high];
    A[high] = temp;

    return partitionLast(A, low, high);
}

void quickSortRandom(int A[], int low, int high)
{
    if (low < high)
    {
        int pi = partitionRandom(A, low, high);
        quickSortRandom(A, low, pi - 1);
        quickSortRandom(A, pi + 1, high);
    }
}

int main()
{
    int n;

    srand(time(0));

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int A[n];
    printf("Enter elements:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &A[i]);

    quickSortRandom(A, 0, n - 1);

    printf("Sorted array (Random Pivot):\n");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);

    return 0;
}
```

Output:



```
C:\Users\kavin\OneDrive\Desktop\DAA\WEEK - 5>gcc quicksort_randomelementaspiviot.c

C:\Users\kavin\OneDrive\Desktop\DAA\WEEK - 5>a
Enter number of elements: 5
Enter elements:
7 1 2 8 5
Sorted array (Random Pivot):
1 2 5 7 8
```

Analysis:

The Random Pivot Quick Sort is better.

Random pivot is better because it reduces the chance of getting the worst-case time complexity.

If we always choose the first or last element as pivot, then for already sorted or reverse sorted arrays, Quick Sort becomes very slow $O(n^2)$.

But if we choose the pivot randomly, the partitions are more balanced in most cases.

Balanced partition means fewer comparisons and faster sorting.

So, Random Pivot Quick Sort gives better performance on average and avoids predictable worst cases.