Date : 27-08-2025

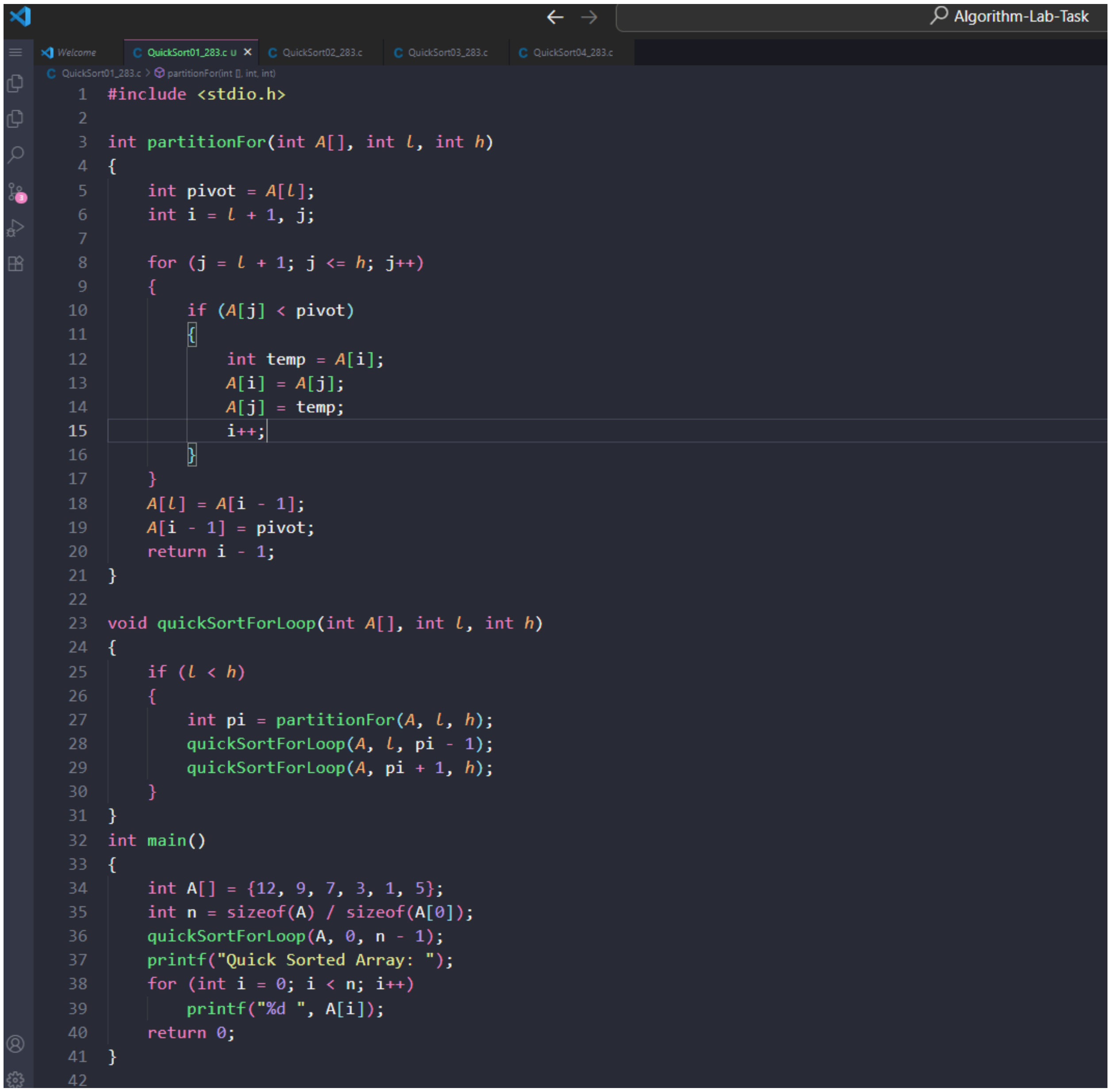
Name : Amit Sarkar

Intake : 51

ID : 22235103283

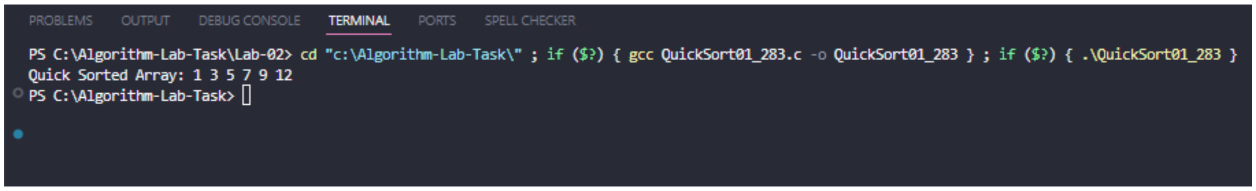
## Lab Report 02

1. ***Take the pivot element as the first element of the array. Using For Loop.***



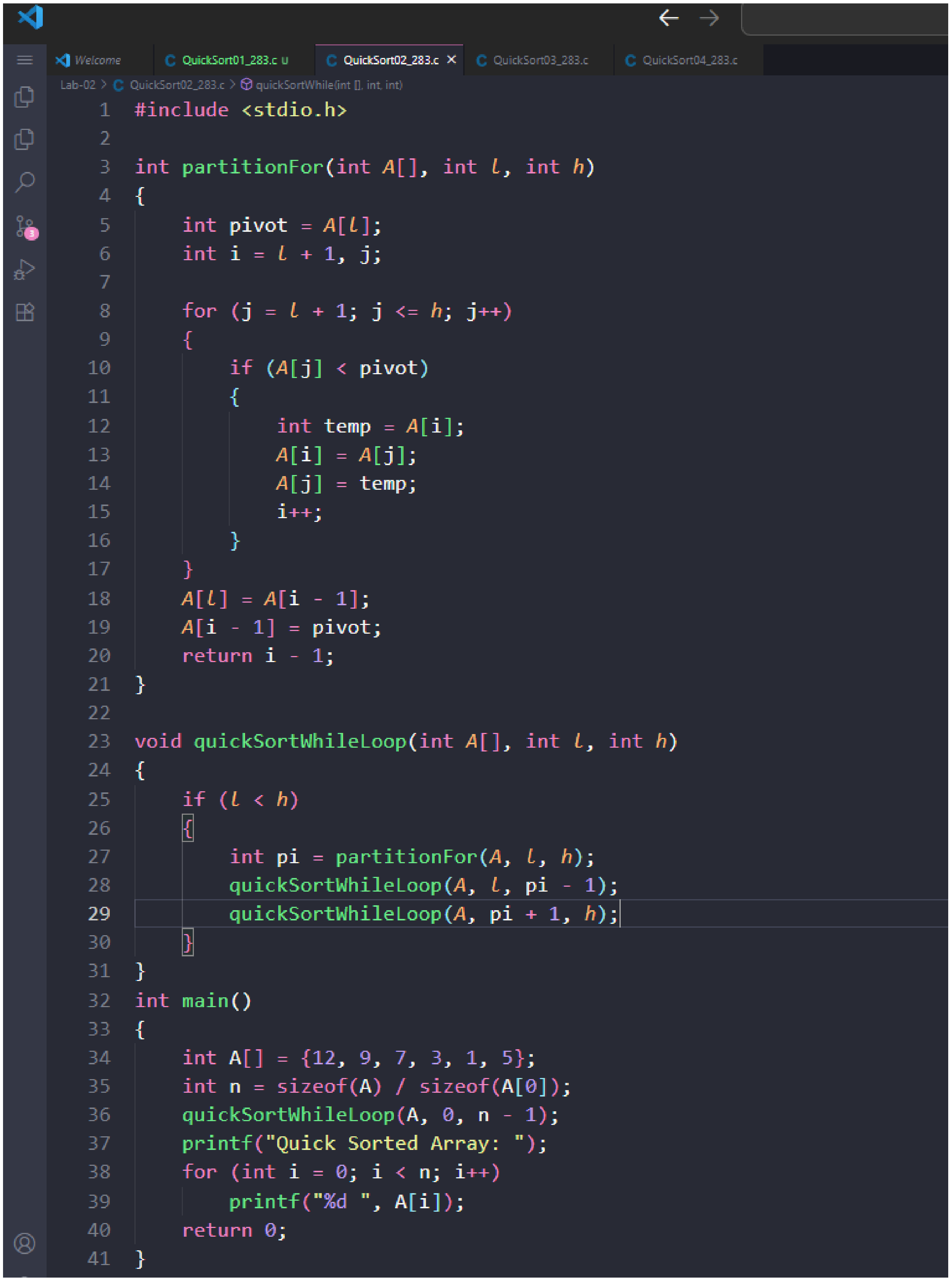
```c
#include <stdio.h>

int partitionFor(int A[], int l, int h)
{
    int pivot = A[l];
    int i = l + 1, j;

    for (j = l + 1; j <= h; j++)
    {
        if (A[j] < pivot)
        {
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
            i++;
        }
    }
    A[l] = A[i - 1];
    A[i - 1] = pivot;
    return i - 1;
}

void quickSortForLoop(int A[], int l, int h)
{
    if (l < h)
    {
        int pi = partitionFor(A, l, h);
        quickSortForLoop(A, l, pi - 1);
        quickSortForLoop(A, pi + 1, h);
    }
}
int main()
{
    int A[] = {12, 9, 7, 3, 1, 5};
    int n = sizeof(A) / sizeof(A[0]);
    quickSortForLoop(A, 0, n - 1);
    printf("Quick Sorted Array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);
    return 0;
}

```

**OutPut :**



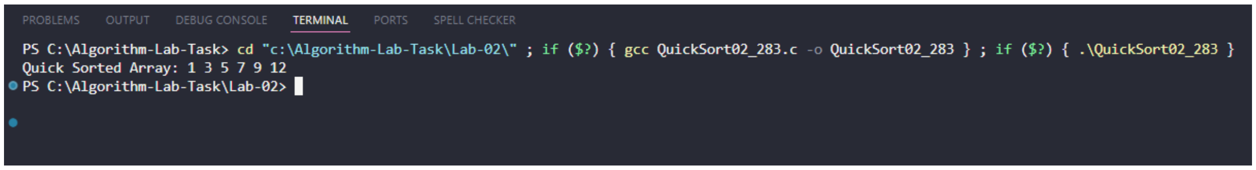
```
PS C:\Algorithm-Lab-Task\Lab-02> cd "c:\Algorithm-Lab-Task\" ; if ($?) { gcc QuickSort01_283.c -o QuickSort01_283 } ; if ($?) { .\QuickSort01_283 }
Quick Sorted Array: 1 3 5 7 9 12
PS C:\Algorithm-Lab-Task>
```

***2. Take the pivot element as the first element of the array. Using While Loop.***



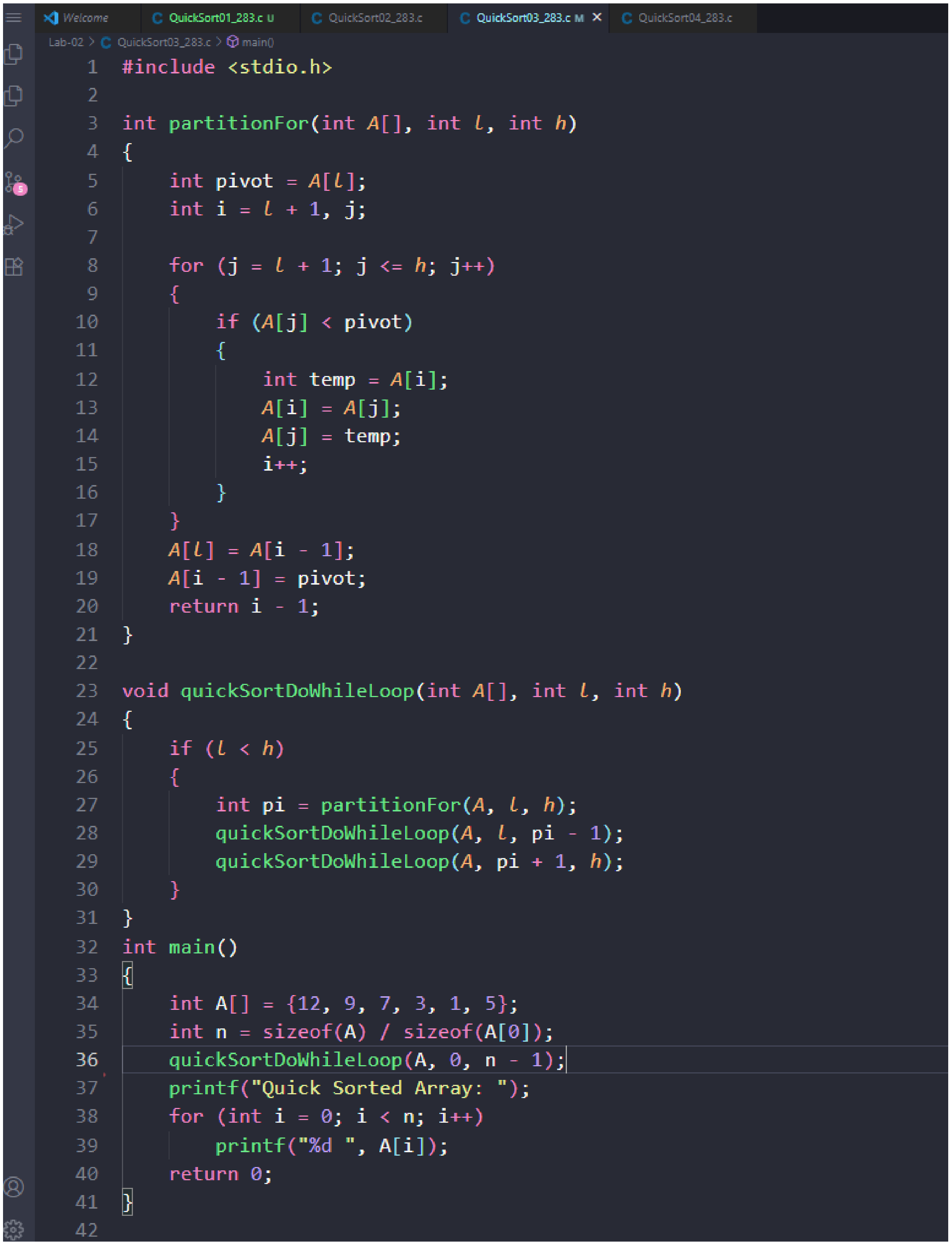
```c
#include <stdio.h>

int partitionFor(int A[], int l, int h)
{
    int pivot = A[l];
    int i = l + 1, j;

    for (j = l + 1; j <= h; j++)
    {
        if (A[j] < pivot)
        {
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
            i++;
        }
    }
    A[l] = A[i - 1];
    A[i - 1] = pivot;
    return i - 1;
}

void quickSortWhileLoop(int A[], int l, int h)
{
    if (l < h)
    {
        int pi = partitionFor(A, l, h);
        quickSortWhileLoop(A, l, pi - 1);
        quickSortWhileLoop(A, pi + 1, h);
    }
}
int main()
{
    int A[] = {12, 9, 7, 3, 1, 5};
    int n = sizeof(A) / sizeof(A[0]);
    quickSortWhileLoop(A, 0, n - 1);
    printf("Quick Sorted Array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);
    return 0;
}
```

***OutPut :***



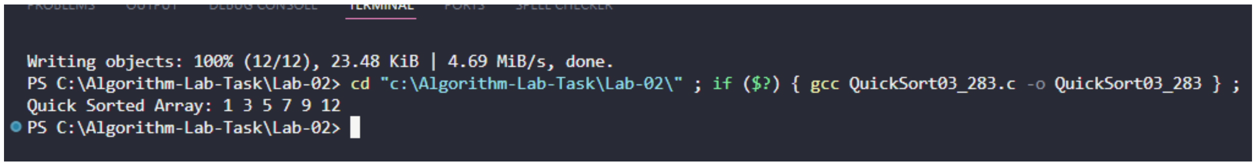
```
PS C:\Algorithm-Lab-Task> cd "c:\Algorithm-Lab-Task\Lab-02\" ; if ($?) { gcc QuickSort02_283.c -o QuickSort02_283 } ; if ($?) { .\QuickSort02_283 }
Quick Sorted Array: 1 3 5 7 9 12
PS C:\Algorithm-Lab-Task\Lab-02>
```

3. ***Take the pivot element as the first element of the array. Using Do While***



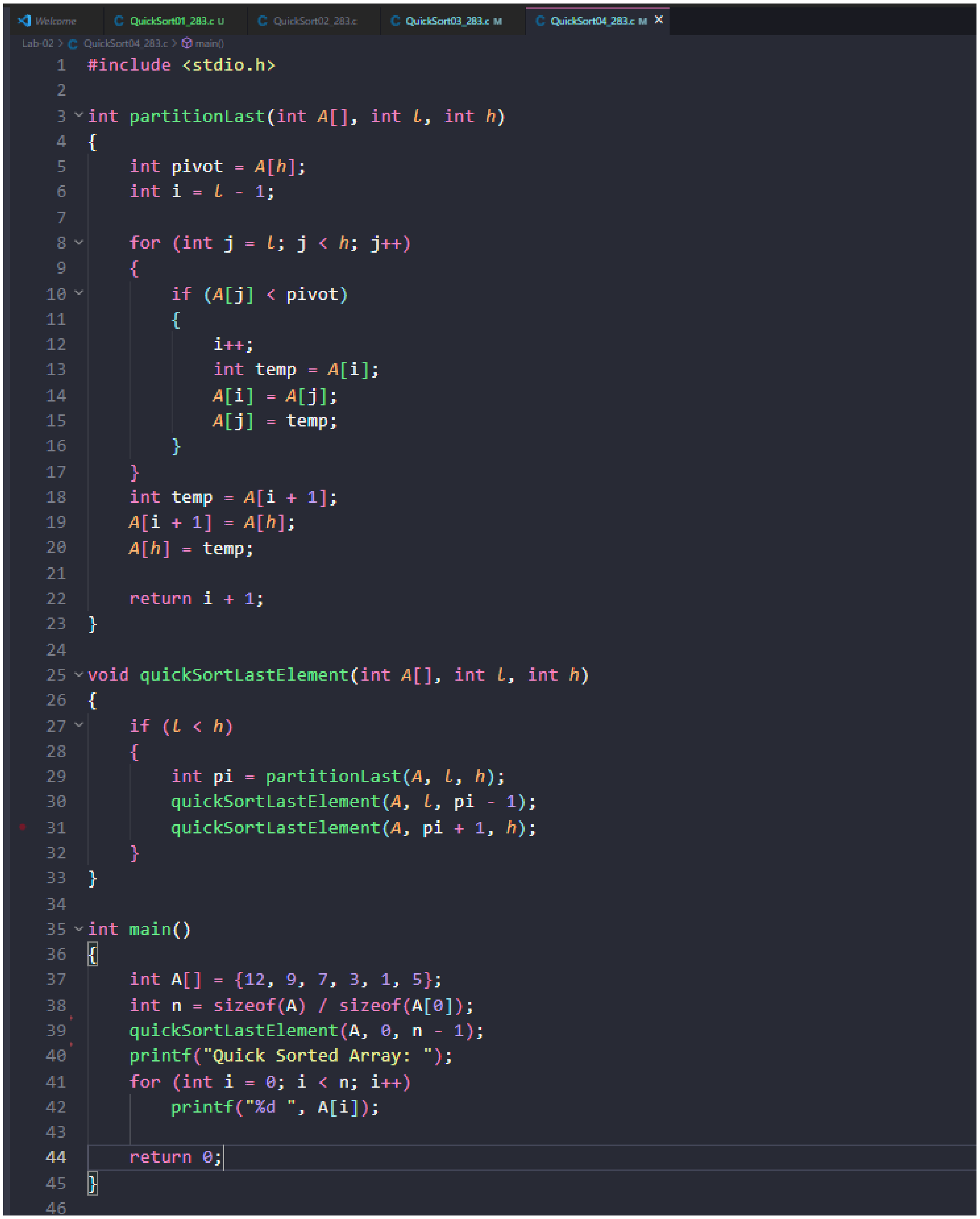
```c
#include <stdio.h>

int partitionFor(int A[], int l, int h)
{
    int pivot = A[l];
    int i = l + 1, j;

    for (j = l + 1; j <= h; j++)
    {
        if (A[j] < pivot)
        {
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;
            i++;
        }
    }
    A[l] = A[i - 1];
    A[i - 1] = pivot;
    return i - 1;
}

void quickSortDoWhileLoop(int A[], int l, int h)
{
    if (l < h)
    {
        int pi = partitionFor(A, l, h);
        quickSortDoWhileLoop(A, l, pi - 1);
        quickSortDoWhileLoop(A, pi + 1, h);
    }
}
int main()
{
    int A[] = {12, 9, 7, 3, 1, 5};
    int n = sizeof(A) / sizeof(A[0]);
    quickSortDoWhileLoop(A, 0, n - 1);
    printf("Quick Sorted Array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);
    return 0;
}
```

***OutPut :***



```
Writing objects: 100% (12/12), 23.48 KiB | 4.69 MiB/s, done.
PS C:\Algorithm-Lab-Task\Lab-02> cd "c:\Algorithm-Lab-Task\Lab-02\" ; if ($?) { gcc QuickSort03_283.c -o QuickSort03_283 } ;
Quick Sorted Array: 1 3 5 7 9 12
PS C:\Algorithm-Lab-Task\Lab-02>
```

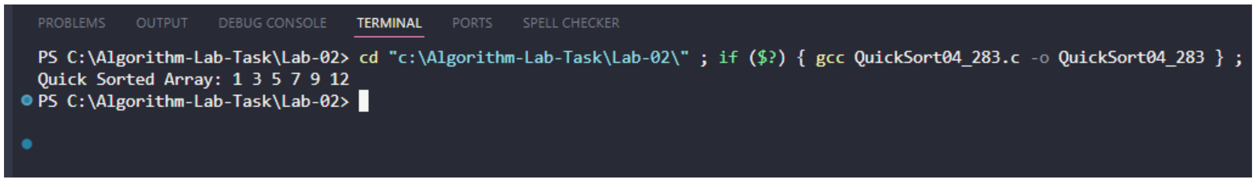
- ***4. Take the pivot element as the Last element of the array. Using For Loop***


Welcome | QuickSort01_283.c U | QuickSort02_283.c | QuickSort03_283.c M | QuickSort04_283.c M

Lab-02 > QuickSort04_283.c > main()

```c
#include <stdio.h>

int partitionLast(int A[], int l, int h)
{
    int pivot = A[h];
    int i = l - 1;

    for (int j = l; j < h; 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[h];
    A[h] = temp;

    return i + 1;
}

void quickSortLastElement(int A[], int l, int h)
{
    if (l < h)
    {
        int pi = partitionLast(A, l, h);
        quickSortLastElement(A, l, pi - 1);
        quickSortLastElement(A, pi + 1, h);
    }
}

int main()
{
    int A[] = {12, 9, 7, 3, 1, 5};
    int n = sizeof(A) / sizeof(A[0]);
    quickSortLastElement(A, 0, n - 1);
    printf("Quick Sorted Array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);

    return 0;
}

```

***OutPut :***



```
PS C:\Algorithm-Lab-Task\Lab-02> cd "c:\Algorithm-Lab-Task\Lab-02\" ; if ($?) { gcc QuickSort04_283.c -o QuickSort04_283 } ;
Quick Sorted Array: 1 3 5 7 9 12
PS C:\Algorithm-Lab-Task\Lab-02>
```