Quicksort

Problem Statement

Use the `In_Place_Partition` function from ProgHW-I to implement the *randomized Quicksort* algorithm, which takes as input an array `A` and two indexes `left` and `right`.
The algorithm proceeds as follows. **The pseudo code is updated (9/19)**

```
function recursive_quick_sort(int A[], int left, int right)
{
    1. If right - left < 1, then return.

    2. Pick a random index from [left...right], say, i.

    3. key <-- In_Place_Partition(A, left, right, i);

    4. recursive_quick_sort(A, left, key-1);
       recursive_quick_sort(A, key+1, right);
}
```

In addition, implement the following function:

```
function quick_sort(int A[], int n)
{
    // Start the recursive sorting
    recursive_quick_sort(A, 0, n-1);
}
```

Submission Instructions

This is a function implementation task. Your submitted code must include the following identifier:

```
/* probID: W3-A-QuickSort */
```

and must include the implementation of the `quick_sort` function (additional function declarations/implementations are allowed if necessary), but must not include a `main` function, as its presence will cause compilation errors.

When submitting, choose the language `C++ - function only`.

On the server side, the DomJudge system will compile and test your submitted code along with the following C++ code:

```
#include <stdio.h>

void quick_sort(int[], int);

int A[200000], n;

int main()
{
    scanf("%d", &n);
    for(int i = 0; i < n; i++)
        scanf("%d", &A[i]);

    quick_sort(A, n);

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

    return 0;
}
```

Example

Input:

```
5
4 3 2 1 5
```

Output:

```
1 2 3 4 5
```