Quick Select

Problem Statement

Implement the recursive function `quick_select` to find the k-th smallest element in a given array `A`.

```
int quick_select(int A[], int left, int right, int k);
```

The algorithm proceeds as follows, where we use the `In_Place_Partition` function from ProgHW-I.

```
function quick_select(int A[], int left, int right, int k)
{
    1. If left == right
       return A[left].

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

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

    4. if A[key] is k-th smallest element, i.e., key == left+k-1,
          then return A[key].

    5. if the k-th smallest element is in A[left ~ key-1],
          then recurse on A[left...key-1].
       otherwise, // the k-th smallest element is in A[key + 1 ~ right],
          then recurse on A[key+1...right]
}
```

Submission Instructions

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

```
/* probID: W5-A-Quick-Select */
```

and must include the implementation of the `quick_select` 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>

int quick_select(int[], int, int, int);

int A[10000000];

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

    int value  = quick_select(A, 0, n - 1, k);
    printf("%d\n", value);

    return 0;
}
```

Example

Input1:

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

Output1:

```
2
```

Input2:

```
7
10 4 3 2 5 8 6
3
```

Output2:

```
4
```