159 more points to get your next star!

Rank: **19541** | Points: **2041/2200**

Find the Median ☆

Problem Submissions Leaderboard

RATE THIS CHALLENGE

★ ☆ ☆ ☆ ☆

The median of a list of numbers is essentially it's middle element after sorting. The same number of elements occur after it as before. Given a list of numbers with an odd number of elements, can you find the median?

For example, the median of $arr = [1, 2, 3, 4, 5]$ is 3, the middle element in the sorted array.

Function Description

Complete the findMedian function in the editor below. It must return an integer that represents the median of the array.

findMedian has the following parameter(s):

- arr: an unsorted array of integers

Input Format

The first line contains the integer n, the size of arr.
The second line contains n space-separated integers $arr[i]$

Constraints

- $1 \leq n \leq 1000001$
- n is odd
- $-10000 \leq arr[i] \leq 10000$

Output Format

Output one integer, the median.

Sample Input 0

```
7
0 1 2 4 6 5 3
```

Sample Output 0

```
3
```

Explanation 0

The sorted $arr = [0, 1, 2, 3, 4, 5, 6]$. It's middle element is at $arr[3] = 3$.

C++14

```
    std::swap(arr[pivotIndex], arr[right]); // Move pivot to end
```

```cpp
  auto storeIndex = left;
  for (int i = left; i < right; ++i)
    if (arr[i] < pivotValue) {
      std::swap(arr[storeIndex], arr[i]);
      ++storeIndex;
    }
  std::swap(arr[right], arr[storeIndex]); // Move pivot to its final place
  return storeIndex;
}

int quickSelect(std::vector<int> &arr, int left, int right, int k) {
  if (left == right)  // If the list contains only one element,
    return arr[left]; // return that element
  // select a pivotIndex between left and right,
  auto pivotIndex = left + std::floor(rand() % (right - left + 1));
  // e.g., left + floor(rand() % (right - left + 1))
  pivotIndex = partition(arr, left, right, pivotIndex);
  // The pivot is in its final sorted position
  if (k == pivotIndex)
    return arr[k];
  else if (k < pivotIndex)
    return quickSelect(arr, left, pivotIndex - 1, k);
  else
    return quickSelect(arr, pivotIndex + 1, right, k);
}

int main() {
  std::size_t n{0};
  std::cin >> n;
  std::vector<int> arr(n, 0);
  for (auto &element : arr)
    std::cin >> element;

  std::cout << quickSelect(arr, 0, n - 1, n / 2) << '\n';

  return 0;
}
```

Line: 5 Col: 18

Upload Code as File   ☐ Test against custom input   Run Code   Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Test case 0 ✓

Test case 1 ✓

Test case 2 ✓

Test case 3 ✓

Compiler Message

```
Success
```

Input (stdin) Download

```
7
0 1 2 4 6 5 3
```

Expected Output Download

```
3
```