DSA Problem

Problem:
Given an array of integers, return the **second largest unique number** in the array.
If it doesn’t exist, return -1.

Example:

Input: [3, 5, 2, 5, 6, 6, 1]

Output: 5

Input: [7, 7, 7]

Output: -1

Solution:

- Code in C++
- Clean, readable code with time complexity better than $O(n^2)$

Code with comments:

```
#include<bits/stdc++.h>

using namespace std;

int getSecondLargest(vector<int> &arr) {

  // initializing largest and secondLargest both set to -1.

  int largest = -1;

  int secondLargest = -1;


  for (int i = 0; i < arr.size(); i++) {

    /*

    If the current element is greater than largest, we update secondLargest to the old
largest, and update largest to this new value.

    */
```

```
        if(arr[i] > largest) {
                    secondLargest = largest;
            largest = arr[i];
        }
        /*
        if the current element is smaller than largest but greater than secondLargest, then
we update secondLargest to this value.
        */
       else if(arr[i] < largest && arr[i] > secondLargest) {
                    secondLargest = arr[i];
        }
    }
    return secondLargest;
}

int main() {
    vector<int> arr = {3, 5, 2, 5, 6, 6, 1};
    cout << getSecondLargest(arr);
    return 0;
}
```