|  |  |
| --- | --- |
| **Problem Statement Title** | Finding Nearest Primes |
| **C programming Concept** | Pointer arithmetic, Array – 1D |
| **Additional Programming Concepts** | Sorting |

**Introduction:**

Professor Jsmith amazed at the problem-solving ability among a group of students, so he starts the session daily with new challenges to help them gain expertise. Today's challenge is to find prime numbers using the following rule:

* The students were given an array of integers, and their task was to create a special function called "squareNumbers" that could modify the array in place, replacing each element with its square using **pointer arithmetic.**
* After modifying the array,Calculate the difference(diff) between the largest and smallest squared numbers in the list. If diff is a prime number, then return the diff value. Or

If the diff is not prime number calculate the nearest next prime number.

**Note: The result(next prime number) should be within the range of INT\_MIN to INT\_MAX.**

**Function Details:**

* Function name: squareNumbers
* Arguments/input:
  + Input 1: A pointer to an array of integers (int\* arr)
  + Input 2: Array Length (int size)
  + Return value/Output: Integer, the function returns the prime number as per the above rule.

**Example 1:**

* Input 1: {2, 5, 1, 6}
* Input 2: 4
* Expected Output: 37
* Explanation:
  + Step 1: Input array is {2,5,1,6}
  + Step 2: Each element in the array is squared, resulting in the modified array {4, 25, 1, 36}.
  + Step 3: The smallest element in the modified array is 1, and the largest element is 36. Their difference is calculated as 36 - 1 = 35.
  + Step 4: Since 35 is not a prime number, the function finds the nearest next prime number to 35, which is 37. Therefore, the function returns 37 as the result.