1.

1. Prove that $\frac{1}{n^2}$ is $O(1/n)$

By this definition:

$f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

$= \frac{1}{n^2} \leq c \cdot \frac{1}{n}$

$n^2 \times \frac{1}{n^2} \leq \frac{c}{n} \times n^2 n$

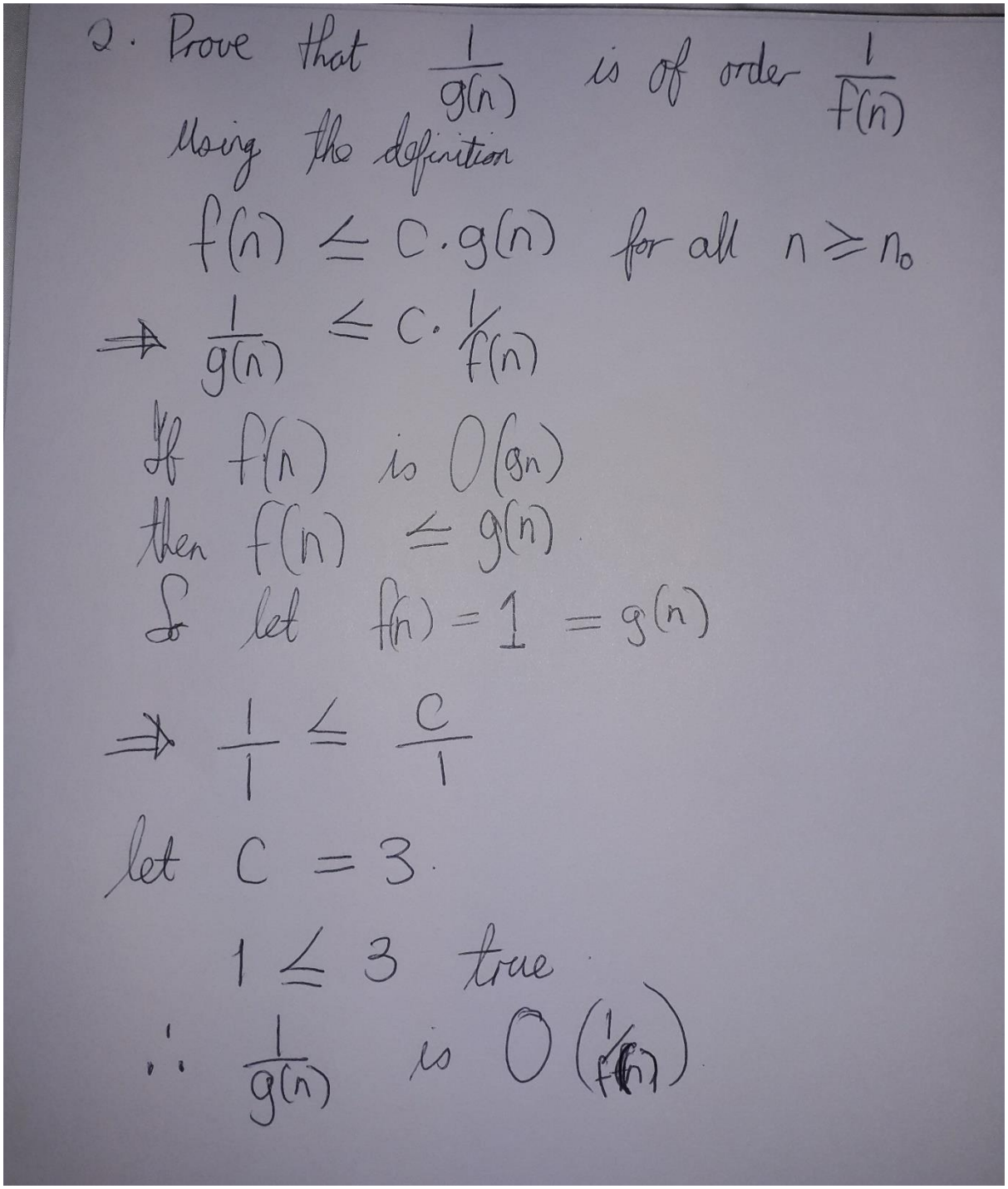
$1 \leq c \cdot n$

let $c = 2$ & $n_0 = 1$

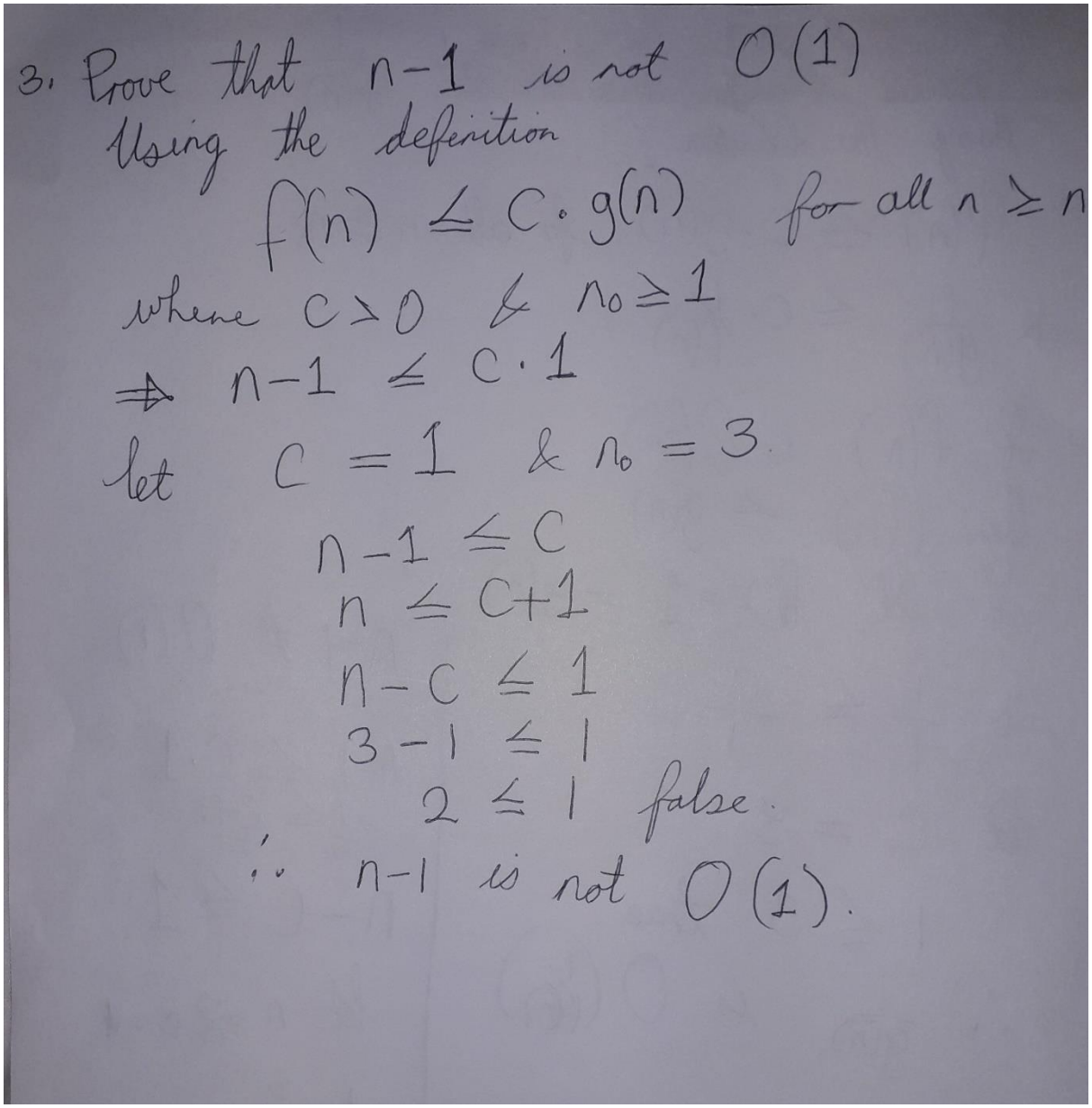
$\therefore 1 \leq 2$ true.

$\therefore \frac{1}{n^2}$ is $O(1/n)$

2.


2. Prove that $\frac{1}{g(n)}$ is of order $\frac{1}{f(n)}$
Using the definition
$f(n) \leq C \cdot g(n)$ for all $n \geq n_0$
$\Rightarrow \frac{1}{g(n)} \leq C \cdot \frac{1}{f(n)}$
If $f(n)$ is $O(gn)$
then $f(n) \leq g(n)$
So let $f(n) = 1 = g(n)$
$\Rightarrow \frac{1}{1} \leq \frac{C}{1}$
let $C = 3$.
$1 \leq 3$ true
$\therefore \frac{1}{g(n)}$ is $O(\frac{1}{f(n)})$

3.



4.

```
public int most_times(int[] A, int n) {
	int toReturn = -1;
	HashMap<Integer, Integer> tracker = new HashMap<Integer, Integer>();
	Arrays.sort(A);

	int largest = -1;
	for (int elem : A) {
		if (tracker.containsKey(elem)) {
			tracker.replace(elem, 1 + tracker.get(elem));
		} else {
```

```
                tracker.put(elem, 1);
            }
            if (largest < tracker.get(elem)) {
                largest = tracker.get(elem);
                toReturn = elem;
            }
        }
        return toReturn;
    }
```

5.

The algorithm will not terminate if ***x*** is not in the array because whenever ***i*** is greater than or equal to ***n*** it is made to be 1 again. Therefore, the condition in the while loop (***i<n***) never evaluates to false.

For example, using the array below where n=10, the first iteration goes through the even indexes shown in blue.

5	**1**	**2**	**3**	**5**	**20**	**36**	**78**	**20**	**300**
0	1	2	3	4	5	6	7	8	9

When it gets to the 8th index, and ***x*** has not been found, 2 will be added to 8 which will evaluate to 10 and since 10 meets the condition ≥ ***n-1***, ***i*** will be assigned the value of 1. Now the iteration below in green shall take place and if ***x*** is still not found in ***L*** when ***i*** = 9, then ***i*** will be assigned the value of 1 again since 9 + 2 = 11 and 11 meets the condition ≥ ***n-1***.

5	**1**	**2**	**3**	**5**	**20**	**36**	**78**	**20**	**300**
0	1	2	3	4	5	6	7	8	9

6.

The algorithm always produces the correct output because if the element ***x*** is not in ***L*** the loop shall terminate when ***i*** becomes equal to ***n*** and -1 shall be returned. However, if ***x*** is in ***L,*** it will be found, and its index returned within the loop because just as in the algorithm in Q5 the algorithm shall loop through the entire array first starting with the even numbers and ending with the odd numbers. So, if there are 3

elements in the array, ***j*** shall loop as **0** → **2** → **1** covering all 3 indexes before ***i*** becomes ≥ ***n*** and the loop breaks.