For question 1 proceed as follows:

1. First explain what needs to be proven: "We need to find constants $c > 0$ and $n_0 \geq 1$ integer such that ...".
2. Simplify the above inequality.
3. Determine values for c and n_0 as required.

For question 2, **you must use a proof by contradiction**:

- First give the claim that you will assume true and from which you will derive a contradiction.
- Use the definition of order to write the inequality from which you will derive the contradiction.
- Simplify the inequality and explain how you derive a contradiction from it.

---

1. (3 marks) Let $f(n)$ and $g(n)$ and $h(n)$ be non-negative functions such that $f(n)$ is $O(g(n))$. Use the definition of "big Oh" to prove that $f(n) \times h(n)$ is $O(g(n) \times h(n))$.
2. (3 marks) Use the definition of "big Oh" to prove that n is not $O(1/n)$.
3. (5 marks) Let A be an array storing $n \geq 1$ integer values. We say that A is *symmetric* if either
   - $n = 1$, or
   - $n > 1$ and $A[i] = A[n - i - 1]$, for all $i = 0, \ldots, \lfloor \frac{n}{2} \rfloor - 1$, where $\lfloor \frac{n}{2} \rfloor$ means to round $\frac{n}{2}$ down to the nearest integer, so for example $\lfloor \frac{3}{2} \rfloor = 1$ and $\lfloor \frac{2}{2} \rfloor = 1$.

   Complete the provided Java class Symmetric.java by designing and implement in Java an algorithm `isSymmetric(int[] A,int n)` that decides whether a given input array A of size n is symmetric. You **must** submit the completed Symmetric.java file as part of your assignment.

   If A is symmetric your algorithm must return the value `true`, otherwise it must return the value `false`. For example if A is the following array:

9	7	4	6	1	3	1	6	4	7	9
0	1	2	3	4	5	6	7	8	9	10

   Then, the algorithm must return the value `true`. However, if the array A is as follows:

4	7	3	9	1	1	7	3	7	4
0	1	2	3	4	5	6	7	8	9

Then, the algorithm must return the value `false` as $A[3] = 9 \neq A[10-3-1] = 7$.

4. Consider the following algorithm for the problem of deciding whether all values stored in an array A of size $n > 1$ are different.

   Algorithm areDifferent(A, n)
   Input: Array A storing n integer values.
   Out: `true` if all values in A are different; `false` otherwise.

   for $i \leftarrow 1$ **to** $n-1$ **do**
   for $j \leftarrow 0$ **to** $i-1$ **do**
   if $A[j] = A[i]$ **then return** `false`
   return `true`

   i. Prove that this algorithm is correct by showing the following:
   a. (1 mark) Show that the algorithm terminates.
   b. (3 marks) Show that the algorithm always produces the correct answer.
   First state what needs to be proven: "If all values stored in A are different then the algorithm must ... ". Then show that the algorithm is correct for each one of the two possible outputs.

   ii. (1 mark) Explain what the worst case for the algorithm is.

   iii. (4 marks) Compute the time complexity of the algorithm in the worst case. You must give the order of the time complexity using "big-Oh" notation and you must explain how you computed the time complexity.

5. (2 marks) **Optional question**. Download from OWL the java class `Search.java`, which contains implementations of 3 different algorithms for solving the search problem:

   - `LinearSearch`, of time complexity $O(n)$.
   - `QuadraticSeach`, of time complexity $O(n^2)$.
   - `FactorialSearch`, of time complexity $O(n!)$.

   Modify the `main` method so that it prints the worst case running times of the above algorithms for the following input sizes:

   - `FactorialSearch`, for input sizes $n = 7, 8, 9, 10, 11, 12$.
   - `QuadraticSeach`, for input sizes $n = 5, 10, 100, 1000, 10000$.
   - `LinearSearch` for, input sizes $n = 5, 10, 100, 1000, 10000, 100000$.

   Fill out the following tables indicating the running times of the algorithms for the above input sizes. You do not need to include your code for the Search class.

n	Linear Search
5	
10	
100	
1000	
10000	
100000	

n	Quadratic Search
5	
10	
100	
1000	
10000	

n	Factorial Search
7	
8	
9	
10	
11	
12	