CS 2210 Data Structures and Algorithms
Assignment 1 (20 marks)
Due: September 27 at 11:55 pm.
Important: No late concept assignments will be accepted

Please submit through OWL a pdf file or an image file with your solution to the assignment. You must also submit the completed java class MostTimes.java. You are encouraged to type your answers. If you decide to submit hand-written answers to the questions please make sure that the TA will be able to read your solutions. If the TA cannot read your answers you will not be given credit for them.
Remember that concept assignments must be submitted by the due date; **no late concept assignments will be accepted** unless you have an academic accommodation.

For questions 1 and 2 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 3, **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) Use the definition of "big Oh" to prove that $\frac{1}{n^2}$ is $O(\frac{1}{n})$.
2. (3 marks) Let $f(n)$ and $g(n)$ be non-negative functions such that $f(n)$ is $O(g(n))$. Use the definition of "big Oh" to prove that $\frac{1}{g(n)}$ is $O(\frac{1}{f(n)})$.
3. (3 marks) Use the definition of "big Oh" to prove that $n - 1$ is not $O(1)$.
4. (5 marks) Let A be an array storing n integer values. Complete the provided Java class MostTimes.java by designing and implement in Java an algorithm `most_times(int[] A,int n)` that returns the value in A that appears the largest number of times. If several values appear the largest number of times, your algorithm must select the smallest one.

   For example if A is the following array:

12	1	2	3	7	12	3	56	12	224	448
0	1	2	3	4	5	6	7	8	9	10

   then, the algorithm must return the value 12, which appears 3 times in A. If the array A is as follows:

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

then the algorithm must return the value 5, since 5 and 20 appear in A twice, but $5 < 20$.

5. (*Proof of termination*) Consider the following algorithm for the search problem.

   Algorithm find(L, n, x)
   Input: Arrays L storing $n \geq 2$ integer values and value x.
   Out: Position of x in L, if x is in L, or -1 if x is not in L

   $i \leftarrow 0$
   while $i < n$ **do** {
   if $L[i] = x$ **then return** i
   else $i \leftarrow i + 2$
   if $i \geq n - 1$ **then** $i \leftarrow 1$
   }
   return -1

   Prove that either (i) the algorithm always terminates, or (ii) there are instances for which the algorithm does not terminate.

   Note that to prove that the algorithm terminates you cannot just give an example and show that the algorithm terminates on that example; instead you must prove that when the algorithm is given as input **any** array L storing n integer values and an integer value x, the algorithm will terminate.

   However, to show that the algorithm does not always terminate it is enough to show an example for which you explain why the algorithm does not finish.

6. (*Proof of correct output*) Consider the following algorithm for the search problem.

   Algorithm search(L, n, x)
   Input: Arrays L storing $n \geq 2$ integer values and value x.
   Out: Position of x in L, if x is in L, or -1 if x is not in L

   $i \leftarrow 0$
   $j \leftarrow 0$
   while $i < n$ **do** {
   if $L[j] = x$ **then return** j
   else $j \leftarrow j + 2$
   if $j \geq n - 1$ **then** $j \leftarrow 1$
   $i \leftarrow i + 1$
   }
   return -1

   Note that this algorithm always terminates because initially i has value 0 and in each iteration of the while loop the value of i increases by 1; therefore, the loop cannot perform more than n iterations.

   Prove that either (i) the algorithm always produces the correct output, or (ii) there are instances for which the algorithm does not produce the correct output.

   Note that to prove that the algorithm always produces the correct output you cannot just give an example and show that the algorithm gives the correct output for that example; instead you must prove that when the algorithm is given as input **any** array L storing n integer values and an integer value x, the algorithm will output either the position of x in L, or -1 if x is not in L.

   However, to show that the algorithm does not always produce the correct output it is enough to show an example for which you explain why the algorithm produces an incorrect output.

7. (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	