CS 2210 Data Structures and Algorithms Assignment 1 (20 marks).

Due: January 29 at 11:55 pm.

Important: No late concept assignment will be accepted

Please submit through OWL a pdf file or an image file with your solution to the assignment. You are **strongly 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 integer $n_0 \geq 1$ such that . . .".
2. Simplify the above inequality.
3. Determine values for c and n_0 as required. Do not forget the value for n_0.

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. (2 marks) Use the definition of "big Oh" to prove that $\frac{(3n+4)}{(n^2+2)}$ is O($\frac{1}{n}$)
2. (2 marks) Let $f_1(n), f_2(n), g_1(n)$, and $g_2(n)$ be positive functions. If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$, using the definition of "big Oh" show that $f_1(n) \cdot f_2(n)$ is $O(g_1(n) \cdot g_2(n))$
3. (2 marks) Use the definition of "big Oh" to prove that 4^n is not O(2^n)
4. Consider the following algorithm for the problem of counting the number of palindromes in a given string T of length $n \geq 1$. The function **isPalindrome()** takes a string input and return True if the input string is a palindrome and False otherwise.

Data: String T with n lowercase letters
Result: The number of palindrome substrings in string T

```
c ← 0;
for i ← 0 to n − 1 do
    for j ← i to n − 1 do
        substring ← T[i to j];
        if isPalindrome(substring) then
            c ← c + 1;
        end
    end
end
return c
```

(a) Prove that this algorithm is correct by providing the following:

i. (2 marks) Show that the algorithms terminates.

Note: To prove that the algorithm terminates you cannot just give an example and show that the algorithm terminates on it; instead you must prove that when the algorithm is given any input string of length n, for any value of n, the algorithm will always end.

ii. (2 marks) Show that the algorithm always produces the correct answer.

Note: To prove that the algorithm is correct you cannot just give an example and show that the algorithm computes the correct output. You must prove that when the algorithm is given any input string of length $n \geq 1$, the algorithm correctly outputs the number of palindromes in the string. Assume that the function **isPalindrome()** always produces correct output.

iii. (4 marks) Compute the time complexity of this algorithm in the worst case. You must explain how you computed the time complexity, and you must give the order of the complexity. Assume that the function **isPalindrome()** takes only a constant time to execute.

5. Consider the following algorithm that updates the values in the array A.

(a) (3 marks) Compute the time complexity of this algorithm in the **best case**. You must explain how you computed the time complexity, and you must give the order of the complexity.

(b) (3 marks) Compute the time complexity of this algorithm in the **worst case**. You must explain how you computed the time complexity, and you must give the order of the complexity.

Data: Array A storing n integer values
Result: None

```
i ← 0;
while i < n do
    if i = 0 then
        step ← 1
    end
    else if A[i] > 0 then
        A[i] ← −A[i];
        step ← −1;
    end
    i ← i + step;
end
```
