Name(s) ____________________________ CSE Login ____________________________

**Instructions** Follow all instructions *carefully*, failure to do so may result in points being deducted. Clearly label each problem and submit the answers in order. Print out a copy of this cover sheet and staple it to the front of your assignment for grading. Be sure to show sufficient work to justify your answers. When asked to prove something, you must give a formal, rigorous, and complete proof. The CSE academic dishonesty policy is in effect (see [https://cse.unl.edu/academic-integrity](https://cse.unl.edu/academic-integrity)). You are highly encouraged to typeset your assignment using \TeX; if your answers are not legible, you may be required to use \TeX in future assignments.

**Partner Policy** You have the option of working in *pairs* for this assignment. You always have the option of working alone if you chose. If you choose to work as a pair, you must adhere to these guidelines:

1. You must work on *all* problems *together*. You may not simply partition the work between you.

2. You should not discuss problems with other groups or individuals beyond general questions.

3. Hand in only one hard copy with both of your names and logins. The first author should hand in any soft copies of programs.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>125</td>
<td></td>
</tr>
</tbody>
</table>
1. Consider the following “proof” that

\[ 1^k + 2^k + \cdots + n^k \in O(n^{k+1}) \]

**proof** Let \( f(n) = 1^k + 2^k + \cdots + n^k \) and \( g(n) = n^{k+1} \). Then we have that

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{1^k + 2^k + 3^k + \cdots + n^k}{n^{k+1}}
\]

\[
= \lim_{n \to \infty} \left[ \frac{1^k}{n^{k+1}} + \frac{2^k}{n^{k+1}} + \frac{3^k}{n^{k+1}} + \cdots + \frac{n^k}{n^{k+1}} \right]
\]

\[
= \lim_{n \to \infty} \frac{1^k}{n^{k+1}} + \lim_{n \to \infty} \frac{2^k}{n^{k+1}} + \lim_{n \to \infty} \frac{3^k}{n^{k+1}} + \cdots + \lim_{n \to \infty} \frac{n^k}{n^{k+1}}
\]

\[= 0 + 0 + 0 + \cdots + 0\]

\[= 0\]

Therefore, \( f(n) \in O(g(n)) \).

(a) **3 points** The proof is wrong: rewrite the proof for \( k = 1 \) and what the limit method actually proves.

(b) **3 points** Why is the original proof wrong?

(c) **3 points** Prove (using the definition) that

\[ 1^k + 2^k + \cdots + n^k \in O(n^{k+1}) \]

2. **5 points** An algorithm takes 1.25 ms for an input size \( n = 200 \). For each of the following complexity characterizations, indicate how long the algorithm will take to run on an input size of \( n = 2500 \). Assume that low-order terms are negligible.

(a) linear

(b) \( O(n \log n) \)

(c) quadratic

(d) cubic

(e) exponential

3. **5 points** An algorithm takes 0.75 ms for input size \( n = 100 \). For each of the following complexity characterizations, indicate how large an input size can be if the problem must be solved within 1 minute. Assume that low-order terms are negligible.

(a) linear

(b) \( O(n \log n) \)

(c) quadratic

(d) cubic

(e) exponential

4. For each pair of functions determine if \( f(n) \in \Omega(g(n)) \) or \( f(n) \in \Theta(g(n)) \) or \( f(n) \in O(g(n)) \) and provide a proof as specified.

(a) **10 points** For each of the following, give a proof using the definitions.
1. \( f(n) = \log(n) \), \( g(n) = \log(n+1) \)
2. \( f(n) = n^3 + n \log(n) - n \), \( g(n) = n^4 + n \)
3. \( f(n) = \log(n!), \ g(n) = n \log(n) \)
4. \( f(n) = \log_3(n) \), \( g(n) = \log^2(n) \)
5. \( f(n) = \log(n), \ g(n) = \log(\log(n)) \)

(b) 10 points For each of the following, give a proof using limits.
1. \( f(n) = 2^{2n}, \ g(n) = 3^n \)
2. \( f(n) = 2n + \log(n), \ g(n) = \log^2(50n) \)
3. \( f(n) = n^3 + 45n^2, \ g(n) = n^3 \log(n) \)
4. \( f(n) = 2^n, \ g(n) = 32n \)
5. \( f(n) = n^2, \ g(n) = n \log(n) \)

5. 6 points Let \( f(n) = a_0 n^3 + a_1 n^2 + a_2 n + a_3 \) be a cubic function with non-negative coefficients.
(a) Prove, using the definition that \( f(n) \in O(n^3) \).
(b) Prove, using the limit method that \( f(n) \in \Theta(n^3) \).

6. 6 points Let \( f(n) = a_0 e^{cn} \) be an exponential function where \( a_0 \in \mathbb{R}^+ \) and \( c > 1 \) and let \( g(n) = b_0 d^n \) also be an exponential function where \( b_0 \in \mathbb{R}^+ \) and \( d > c \).
(a) Prove, using the definition that \( f(n) \in O(g(n)) \).
(b) Prove, using the limit method that \( f(n) \in O(g(n)) \).

7. 9 points Order the following functions in increasing order of growth and indicate asymptotic equivalences (functions that are \( \Theta \) of each other). You need not give a formal proof for each. This is one list, not three.

\[ 6n \log(n) + 2n, \left(\frac{3}{2}\right)^n, n^n, \log \log(n), \log^2(n), \frac{1}{n}, 2^{10}, n - n^3 + 6n^5, \frac{n}{\log(n)}, n!, \]
\[ 2^{\log(n)}, 2^n, 2^4n, 4^2n, 3n + \log(n^{100}), \log(n) \log \log(n) \]

8. 5 points Consider the following algorithm that computes whether or not a given integer \( n \) is prime or composite.

input : An integer \( n \)
output: true if \( n \) is prime, false otherwise
for \( k = 2, \ldots, \sqrt{n} \) do
    if \( n \mod k = 0 \) then
        output false ;
    end
end
output true ;

(a) Identify the input, the input size, and the elementary operation.
(b) Analyze the algorithm’s running time with respect to the input size \( n \).
(c) Provide a Big-O characterization of the algorithm. Does this algorithm run in polynomial time with respect to the input size? Why or why not?
Consider the following problem. Two weather stations periodically take temperature data samples. Each data point consists of a pair, \((d, t)\) where \(d\) is the date/time of the sample and \(t\) is the temperature. The two weather stations are not necessarily synchronized, however. The collections of data are not necessarily in chronological order and may or may not contain samples on the same time(s) or the same temperature(s). The problem is to detect data anomalies in which the two stations differ in temperature.

(a) Consider the following algorithm that solves this problem.

\[
\begin{align*}
\text{input} & : \text{Two lists of date/temperature data, } A = \{(d_1, t_1), \ldots, (d_n, t_n)\} \text{ and } B = \{(D_1, T_1), \ldots, (D_n, T_n)\} \\
\text{output} & : \text{All data points that differ.} \\
\text{for } i = 1, \ldots, n \text{ do} \\
\quad \text{for } j = 1, \ldots, n \text{ do} \\
\quad \quad \text{if } d_i = D_j \text{ then} \\
\quad \quad \quad \text{if } t_i \neq T_j \text{ then} \\
\quad \quad \quad \quad \text{output } ((d_i, t_i), (D_j, T_j)) \\
\quad \quad \quad \text{end} \\
\quad \quad \text{end} \\
\text{end}
\end{align*}
\]

Analyze this algorithm using the 5-step process and give an asymptotic characterization of the algorithm.

(b) How well would this algorithm perform in practice? Suppose our machine can process data at 1 million pairs per second. How long would it take to execute this algorithm on the following input sizes?

1. \(n = 100\)
2. \(n = 10,000\)
3. \(n = 1,000,000\)
4. \(n = 10,000,000\)
5. \(n = 1,000,000,000\)

(c) Can you do better? Design a better algorithm and give a full analysis. Indicate how fast your algorithm would run on the machine that can process 1 million pairs per second.

For all exercises asking you to design an algorithm you must:

1. Provide full and clear pseudocode
2. Identify the input
3. Identify the input size
4. Identify the elementary operation
5. Analyze the number of times the elementary operations is performed with respect to the input size
6. Provide an asymptotic analysis of your algorithm

For those using \LaTeX you may find the Algorithm2e package useful for typesetting pseudocode.

10. Let \(A = [a_1, a_2, \ldots, a_n]\) be a collection of integers. A pair \((i, j)\) is called an inversion if \(i < j\) but \(a_i > a_j\). For example, if \(A = [2, 3, 8, 6, 1]\) then the list of inversions is \((2, 1), (3, 1), (8, 6), (8, 1), (6, 1)\). Design an algorithm (provide good pseudocode) that, given a collection of integers \(A\) outputs a list of its inversions. Provide a complete analysis of your algorithm.
11. 10 points Three or more points are co-linear if they lie on the same line in 2-space. Give and analyze an algorithm for the following problem. Given a list of \( n \) points, 
\[ \{(x_1, y_1), \ldots, (x_n, y_n)\} \]
find the maximal number of co-linear points.

12. 10 points Recall that the symmetric difference of two sets \( A, B \) contains all elements that are either in \( A \) or in \( B \) but not in both. Design an algorithm that, given \( A, B \) outputs the symmetric difference, \( A \oplus B \).

13. 10 points Let \( A = \{a_1, \ldots, a_n\} \) be a collection of elements. The number of times an element \( x \) appears in \( A \) is its multiplicity. Design an algorithm to output each element in \( A \) and its multiplicity.

14. 10 points (bonus) A propositional formula on \( n \) variables, \( P(x_1, x_2, \ldots, x_n) \) is satisfiable if there exists an assignment of truth values (true or false) to its variables such that it evaluates to true.

(a) Give an algorithm (pseudocode) that, given a formula \( P \) determines if it is satisfiable or not. Analyze your algorithm.

(b) Suppose that we are given a “free” algorithm \( A \) that, given \( P \) and a partial assignment of truth values (that is, some variables are set to T, others to F, others remain variables) outputs true if the formula is still satisfiable and false if it is not. Further suppose that the cost of this algorithm is constant. Give an algorithm that, given a formula determines if it is satisfiable or not and returns a satisfying assignment. Make use of \( A \) as a subroutine and analyze your algorithm.