OU Academic Integrity Pledge

On my honor I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature: ___________________________ Date: ___________________________
Section I. Definitions of Time and Space Complexity

**Definition of Big O:** Let \( f(n) \) and \( g(n) \) be functions mapping non-negative integers to real numbers. We say that \( f(n) \in O(g(n)) \) if there is a real number \( c > 0 \) and a fixed integer \( n_0 \geq 1 \) such that \( f(n) \leq cg(n) \) for every integer \( n \geq n_0 \).

**Definition of Big \( \Omega \):** Let \( f(n) \) and \( g(n) \) be functions mapping non-negative integers to real numbers. We say that \( f(n) \in \Omega(g(n)) \) if there is a real number \( c > 0 \) and a fixed integer \( n_0 \geq 1 \) such that \( f(n) \geq cg(n) \) for every integer \( n \geq n_0 \).

**Definition of Big \( \Theta \):** Let \( f(n) \) and \( g(n) \) be functions mapping non-negative integers to real numbers. We say that \( f(n) \in \Theta(g(n)) \) if there are real numbers \( c, d > 0 \) and a fixed integer \( n_0 \geq 1 \) such that \( dg(n) \leq f(n) \leq cg(n) \) for every integer \( n \geq n_0 \).

1. (2 points) Big O is typically used in computer science to provide which of the following?
   A. A lower limit on the worst-case running time of an algorithm
   B. An upper limit on the worst-case running time of an algorithm
   C. An upper limit on the best-case running time of an algorithm
   D. A and B
   E. B and C

2. (2 points) Big \( \Omega \) is typically used in computer science to provide which of the following?
   A. A lower limit on the worst-case running time of an algorithm
   B. An upper limit on the worst-case running time of an algorithm
   C. An upper limit on the best-case running time of an algorithm
   D. A and B
   E. B and C

3. (2 points) Big \( \Theta \) is typically used in computer science to provide which of the following?
   A. A lower limit on the worst-case running time of an algorithm
   B. An upper limit on the worst-case running time of an algorithm
   C. An upper limit on the best-case running time of an algorithm
   D. A and B
   E. B and C

4. (2 points) Which is generally considered to be a better time complexity?
   A. \( \Theta(n \log_2 n) \)
   B. \( \Theta(n^2) \)
   C. \( \Theta(n) \)
   D. \( \Theta(2^n) \)
   E. \( \Theta(n^2 \log_2 n) \)

5. (2 points) Which is generally considered to be a better time complexity?
   A. \( \Theta(2^n + \log_2 n) \)
   B. \( \Theta(n^2) \)
   C. \( \Theta(n + n \log_2 n) \)
   D. \( \Theta(n^2 + \log_2 n) \)
   E. \( \Theta(n^2 + n) \)

6. (2 points) A time complexity of \( \Theta(n \log_2 n + \log_2 n + n) \) can be simplified to which?
   A. \( \Theta(n \log_2 n) \)
   B. \( \Theta(n \log_2 n + n) \)
   C. \( \Theta(\log_2 n) \)
   D. \( \Theta(n^2) \)
   E. \( \Theta(n) \)
Section II. Applications of Time and Space Complexity

Consider the following pseudocode:

Algorithm RecursiveFunction (a, b) // a and b are integers
    if (a ≤ 1)
        return b;
    else
        return RecursiveFunction (a-2, a/2);
    endif

7. (2 points) What is the time complexity of the RecursiveFunction pseudocode shown above?
   A. \( \Theta(a) \)
   B. \( \Theta(\log_2 a) \)
   C. \( \Theta(1) \)
   D. \( \Theta(a/2) \)
   E. \( \Theta(a + b) \)

8. (2 points) What is the space complexity of the RecursiveFunction pseudocode shown above?
   A. \( \Theta(a) \)
   B. \( \Theta(\log_2 a) \)
   C. \( \Theta(1) \)
   D. \( \Theta(a/2) \)
   E. \( \Theta(a + b) \)

Consider the following pseudocode:

Algorithm IterativeFunction (a, b) // a and b are integers
    while (a > 0)
        b ← a / 2
        a ← a - 2
    endwhile
    return b;

9. (2 points) What is the time complexity of the IterativeFunction pseudocode shown above?
   A. \( \Theta(a) \)
   B. \( \Theta(\log_2 a) \)
   C. \( \Theta(1) \)
   D. \( \Theta(a/2) \)
   E. \( \Theta(a + b) \)

10. (2 points) What is the space complexity of the IterativeFunction pseudocode shown above?
    A. \( \Theta(a) \)
    B. \( \Theta(\log_2 a) \)
    C. \( \Theta(1) \)
    D. \( \Theta(a/2) \)
    E. \( \Theta(a + b) \)
Consider the following pseudocode:

Algorithm DoubleIterativeFunction (n) // n is an integer
i ← 1
while (i < n) do
    print i / 2
    i ← i + 1
endwhile
j ← 1
while (j < n) do
    print (i + j) / 2
    j ← j + 1
endwhile

11. (2 points) What is the time complexity of the DoubleIterativeFunction pseudocode shown above?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(\log_2(i + j))$

12. (2 points) What is the space complexity of the DoubleIterativeFunction pseudocode shown above?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(\log_2(i + j))$

Consider the following pseudocode:

Algorithm DoubleIterativeFunction2 (n) // n is an integer
i ← 1
while (i < n) do
    print i / 2
    i ← i + 1
endwhile
j ← 1
while (j < i) do // note the change in this line
    print (i + j) / 2
    j ← j + 1
endwhile

13. (2 points) What is the time complexity of the DoubleIterativeFunction2 pseudocode shown above?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(\log_2(i + j))$

14. (2 points) What is the space complexity of the DoubleIterativeFunction2 pseudocode shown above?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(\log_2(i + j))$
Section III. Time and Space Complexity of Searching and Sorting

15. (2 points) Starting from an unsorted array of \( n \) items, what is the time complexity for the most time efficient algorithm for finding the lowest valued item and moving it to the first location of the array? (The resulting order of the other elements is unimportant.)
   
   A. \( \Theta(n) \)
   B. \( \Theta(\log_2 n) \)
   C. \( \Theta(n \log_2 n) \)
   D. \( \Theta(1) \)
   E. \( \Theta(n^2) \)

16. (2 points) Starting from a sorted array of \( n \) items, what is the time complexity for the most time efficient algorithm for finding the lowest valued item and moving it to the first location of the array? (The resulting order of the other elements is unimportant.)
   
   A. \( \Theta(n) \)
   B. \( \Theta(\log_2 n) \)
   C. \( \Theta(n \log_2 n) \)
   D. \( \Theta(1) \)
   E. \( \Theta(n^2) \)

17. (2 points) Starting from an unsorted array of \( n \) items, what is the time complexity for the most time efficient algorithm for finding the two lowest valued items and moving them to the first two locations of the array? (The resulting order of the other elements is unimportant.)
   
   A. \( \Theta(n) \)
   B. \( \Theta(\log_2 n) \)
   C. \( \Theta(n \log_2 n) \)
   D. \( \Theta(1) \)
   E. \( \Theta(n^2) \)

18. (2 points) What is the time complexity of the most time efficient algorithm for performing \( m \) searches for arbitrary items in an unsorted array of \( n \) items (where \( m < n \))?
   
   A. \( \Theta(m) \)
   B. \( \Theta(m \log_2 n) \)
   C. \( \Theta(n \log_2 m) \)
   D. \( \Theta(n \log_2 n) \)
   E. \( \Theta(mn) \)

19. (2 points) What is the time complexity of the most time efficient algorithm for performing \( m \) searches for arbitrary items in a sorted array of \( n \) items (where \( m < n \))?
   
   A. \( \Theta(m) \)
   B. \( \Theta(m \log_2 n) \)
   C. \( \Theta(n \log_2 m) \)
   D. \( \Theta(n \log_2 n) \)
   E. \( \Theta(mn) \)

20. (2 points) What is the time complexity of the most time efficient algorithm for sorting an array of \( n \) arbitrary items and then performing \( m \) searches for arbitrary items in the sorted array (where \( m < n \))?
   
   A. \( \Theta(m) \)
   B. \( \Theta(m \log_2 n) \)
   C. \( \Theta(n \log_2 m) \)
   D. \( \Theta(n \log_2 n) \)
   E. \( \Theta(mn) \)

Exam continues with short answer questions.
Short Answer Question 1: Bucket Sort (10 points)

A. What is the time complexity of Bucket Sort?

B. Explain why Bucket Sort has the time complexity you listed above in Part A.
Short Answer Question 2: Insertion Sort (10 points)

A. What is the time complexity of Insertion Sort?

B. Explain why Insertion Sort has the time complexity you listed above in Part A.
Short Answer Question 3: Shell Sort (10 points)

A. Shell Sort is generally considered to be a more efficient variation of which other sort?

B. Explain why Shell Sort is generally more efficient than the sort you listed above in Part A.
Short Answer Question 4: Quicksort (30 points)

Algorithm Quicksort (A, left, right)
   if (left < right)
      pivotPoint ← floor((left+right)/2) // note central pivot
      i ← left - 1
      j ← right + 1
      do
         do
            i ← i + 1
            while (i < A.size) and (A[i] ≤ A[pivotPoint])
            do
               j ← j - 1
               while (j ≥ i) and (A[j] ≥ A[pivotPoint])
               if (i < j) then swap (A[i], A[j])
               while (i < j)
                  if (i < pivotPoint) then j ← i
                  swap (A[pivotPoint], A[j])
                  Quicksort (A, left, j-1)
                  Quicksort (A, j+1, right)
         endif
   endif

Show the steps followed by the Quicksort algorithm given above in pseudocode when sorting the following array. Draw one figure for each call to Quicksort. You may omit calls where left ≥ right.

<table>
<thead>
<tr>
<th>value</th>
<th>76</th>
<th>24</th>
<th>49</th>
<th>68</th>
<th>40</th>
<th>30</th>
<th>89</th>
<th>88</th>
<th>11</th>
<th>59</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
Additional space to answer Short Answer Question 4