Student Name: _____ Student ID # _____

**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: _____

**Notes Regarding this Examination**

**Open Book(s)** You may consult any printed textbooks in your immediate possession during the course of this examination.

**Open Notes** You may consult any printed notes in your immediate possession during the course of this examination.

**No Electronic Devices Permitted** You may not use any electronic devices during the course of this examination, including but not limited to calculators, computers, and cellular phones. All electronic devices in the student's possession must be turned off and placed out of sight (for example, in the student's own pocket or backpack) for the duration of the examination.

**Violations** Copying another's work, or possession of electronic computing or communication devices in the testing area, is cheating and grounds for penalties in accordance with school policies.

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 Ω:** 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 Θ:** 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) If I need to ensure that my program runs efficiently regardless of input values, I need to be concerned about which of the following?
   A. Big O
   B. Big Ω
   C. Big Θ
   D. A and B
   E. A, B, and C

2. (2 points) If I want to consider how fast my program might run given the right input values, I should be concerned about which of the following?
   A. Big O
   B. Big Ω
   C. Big Θ
   D. A and B
   E. A, B, and C

3. (2 points) If I can characterize my program's runtime as a function of input size to within a constant regardless of input values, I am able to report which of the following?
   A. Big O
   B. Big Ω
   C. Big Θ
   D. A and B
   E. A, 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. $\Theta(2^n)$
   D. $\Theta(\log_2 n)$
   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. $\Theta(n + n^2 \log_2 n)$
   D. $\Theta(n^2 + n^2 \log_2 n)$
   E. $\Theta(n + n \log_2 n)$

6. (2 points) A time complexity of $\Theta(n^2 \log_2 n + \log_2 n + n^2)$ can be simplified to which?
   A. $\Theta(n \log_2 n)$
   B. $\Theta(n^2 \log_2 n)$
   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)  // a is a positive integer
  b ← 100
  if (a ≤ b)
    return a;
  else
    return RecursiveFunction (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. $\Theta(b)$
   E. $\Theta(\log_2 b)$

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. $\Theta(b)$
   E. $\Theta(\log_2 b)$

Consider the following pseudocode:

```
Algorithm IterativeFunction (a)  // a is a positive integer
  b ← 1
  while (b < a)
    b ← b × 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. $\Theta(b)$
   E. $\Theta(\log_2 b)$

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. $\Theta(b)$
    E. $\Theta(\log_2 b)$

Consider the following pseudocode:

```
Algorithm DoubleIterativeFunction (n)  // n is a positive integer
  i ← 1
  while (i < n) do
    print i
    j ← n
    while (j > 0) do
      print j
      j ← j - 2
    endwhile
    i ← i + 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. $\Theta(n \log_2 n)$

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. $\Theta(n \log_2 n)$

Consider the following pseudocode:

```
Algorithm DoubleIterativeFunction2 (n)  // n is a positive integer
  i ← 1
  while (i < n) do
    print i
    j ← n
    while (j > 0) do
      print j
      j ← j / 2    // note the change in this line
    endwhile
    i ← i + 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. $\Theta(n \log_2 n)$

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. $\Theta(n \log_2 n)$

Section III. Complexity of Array Operations

15. (2 points) Starting from **two sorted** arrays with a combined total of $n$ items, what is the **time complexity** for the most time-efficient algorithm for **combining** them into a new sorted array of those same $n$ items?
    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 **two unsorted** arrays with a combined total of $n$ items, what is the **time complexity** for the most time-efficient algorithm for **combining** them into a new sorted array of those same $n$ items?
    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 **two sorted** arrays with a combined total of $n$ items, what is the **time complexity** for the most time-efficient algorithm for **finding** the lowest valued item from either array?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(n \log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

18. (2 points) Starting from **two unsorted** arrays with a combined total of $n$ items, what is the **time complexity** for the most time-efficient algorithm for **finding** the lowest valued item from either array?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(n \log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

19. (2 points) Starting from **one unsorted** array of $n$ items, what is the **space complexity** of the **additional** space needed for the most space-efficient algorithm for **sorting** the array?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(n \log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

20. (2 points) Starting from **one sorted** array of $n$ items, what is the **space complexity** of the most space- and time-efficient algorithm for **finding** an arbitrary key value in the array?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(n \log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

Exam continues with short answer questions.

**Short Answer Question 1:** Straight Insertion Sort (10 points)

A. What is the Big O time complexity of Straight Insertion Sort?

B. Explain why Straight Insertion Sort has the time complexity you listed above in Part A.

**Short Answer Question 2:** Straight Insertion Sort (10 points)

A. What is the Big Ω time complexity of Straight Insertion Sort?

B. Explain why Straight Insertion Sort has the Big Ω time complexity you listed above in Part A.

**Short Answer Question 3:** Quick Sort (10 points)

A. What is the Big O time complexity of Quicksort using the first (or last) element of the array to sort as the pivot point?

B. Explain why Quicksort has the Big O time complexity you listed above in Part A.

**Short Answer Question 4:** Quicksort (30 points)

```
Algorithm Quicksort (A, left, right)
  if (left < right)
    pivotPoint ← ⌊(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
```

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` $\not<$ `right`.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| value | 84 | 27 | 59 | 31 | 78 | 73 | 30 | 38 | 39 | 37 |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Additional space to answer Short Answer Question 4