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

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

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

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

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

6. (2 points) A time complexity of $\Theta(n \log_2 n + \log_2 n + n)$ can be simplified to which?
   A. $\Theta(n)$
   B. $\Theta(n \log_2 n)$
   C. $\Theta(n \log_2 n + n)$
   D. $\Theta(\log_2 n)$
   E. $\Theta(n^2)$

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. $\Theta(a+b)$
   B. $\Theta(a)$
   C. $\Theta(\log_2 a)$
   D. $\Theta(1)$
   E. $\Theta(a/2)$

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

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. $\Theta(a+b)$
   B. $\Theta(a)$
   C. $\Theta(\log_2 a)$
   D. $\Theta(1)$
   E. $\Theta(a/2)$

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

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. $\Theta(\log_2(i+j))$
    B. $\Theta(n)$
    C. $\Theta(\log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

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

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. $\Theta(\log_2(i+j))$
    B. $\Theta(n)$
    C. $\Theta(\log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

14. (2 points) What is the space complexity of the `DoubleIterativeFunction2` pseudocode shown above?
    A. $\Theta(\log_2(i+j))$
    B. $\Theta(n)$
    C. $\Theta(\log_2 n)$
    D. $\Theta(1)$
    E. $\Theta(n^2)$

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. $\Theta(n^2)$
    B. $\Theta(n)$
    C. $\Theta(\log_2 n)$
    D. $\Theta(n \log_2 n)$
    E. $\Theta(1)$

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. $\Theta(n^2)$
    B. $\Theta(n)$
    C. $\Theta(\log_2 n)$
    D. $\Theta(n \log_2 n)$
    E. $\Theta(1)$

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 item**s** and moving **them** to the first **two** location**s** of the array? (The resulting order of the other elements is unimportant.)
    A. $\Theta(n^2)$
    B. $\Theta(n)$
    C. $\Theta(\log_2 n)$
    D. $\Theta(n \log_2 n)$
    E. $\Theta(1)$

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

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

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 $\boldsymbol{m}$ **searches** for arbitrary items in the **sorted** array (where $m < n$)?
    A. $\Theta(mn)$
    B. $\Theta(m)$
    C. $\Theta(m \log_2 n)$
    D. $\Theta(n \log_2 m)$
    E. $\Theta(n \log_2 n)$

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 ← ⌊(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	77	25	50	69	41	31	90	89	12	60
index	0	1	2	3	4	5	6	7	8	9

Additional space to answer Short Answer Question 4.