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 $n \geq n_0$.

Note: For questions in this section, consider an algorithm that operates on n data items. It takes 1000 operations to initialize, regardless of the number of data items, then takes $3n^2$ operations to complete.

1. (2 points) Which of the following equations correctly describes the runtime of this algorithm?
   A. $f(n) = 1000$
   B. $f(n) = n^2$
   C. $f(n) = 3n^2$
   D. $f(n) = n^2 + 1000$
   E. $f(n) = 3n^2 + 1000$

2. (2 points) Which of the following equivalence classes would be valid to describe its Big O runtime?
   A. n
   B. n^2
   C. n^3
   D. A and B
   E. B and C

3. (2 points) Which of the following equivalence classes would be *best* to describe its Big O runtime?
   A. n
   B. n^2
   C. n^3
   D. A and B
   E. B and C

4. (2 points) Which of the following equivalence classes would be valid to describe its Big Ω runtime?
   A. n
   B. n^2
   C. n^3
   D. A and B
   E. B and C

5. (2 points) Which of the following equivalence classes would be *best* to describe its Big Ω runtime?
   A. n
   B. n^2
   C. n^3
   D. A and B
   E. B and C

6. (2 points) Which of the following equivalence classes would be valid to describe its Big Θ runtime?
   A. n
   B. n^2
   C. n^3
   D. A and B
   E. B and C

7. (2 points) Which of the following equivalence classes would be *best* to describe its Big Θ runtime?
   A. n
   B. n^2
   C. n^3
   D. A and B
   E. B and C

8. (2 points) If $g(n) = n^2$ which of the following would be valid for c when describing its Big O runtime?
   A. 1
   B. 2
   C. 3
   D. 4
   E. n^2

9. (2 points) If $g(n) = n^2$, and the value of c must be one of the options given above, which of the following would be valid for n_0 when describing its Big O runtime?
   A. 10
   B. 20
   C. 30
   D. 40
   E. n^2

Section II. Time and Space Complexity in Resizable Arrays

10. (2 points) Which of the following describes the runtime for adding n items to an resizable array that doubles in size when full?
    A. $\Theta(1)$
    B. $\Theta(n)$
    C. $\Theta(n \log n)$
    D. $\Theta(n^2)$
    E. $\Theta(n^2 \log n)$

11. (2 points) Which of the following describes the runtime for adding n items to an resizable array that adds space for a constant number of new items when full?
    A. $\Theta(1)$
    B. $\Theta(n)$
    C. $\Theta(n \log n)$
    D. $\Theta(n^2)$
    E. $\Theta(n^2 \log n)$

12. (2 points) Which of the following describes the runtime for adding n items to an resizable array that doubles in size when full, while ensuring that the array is always maintained in order (as determined by some ordering criterion)?
    A. $\Theta(1)$
    B. $\Theta(n)$
    C. $\Theta(n \log n)$
    D. $\Theta(n^2)$
    E. $\Theta(n^2 \log n)$

Section III. Applications of Time and Space Complexity

Consider the following pseudocode (noting that `%` is modulus division, which returns the remainder after integer division):

```
Algorithm RecursiveFunction (a)  // a is a non-negative integer
  if (a = 0)
    return 0
  else
    return (a % 2) + 10 * RecursiveFunction(a / 2)
  endif
```

13. (2 points) What is the time complexity of the `RecursiveFunction` pseudocode shown above?
    A. $\Theta(a)$
    B. $\Theta(\log a)$
    C. $\Theta(1)$
    D. $\Theta(a/2)$
    E. $\Theta(a\%2)$

14. (2 points) What is the space complexity of the `RecursiveFunction` pseudocode shown above?
    A. $\Theta(a)$
    B. $\Theta(\log a)$
    C. $\Theta(1)$
    D. $\Theta(a/2)$
    E. $\Theta(a\%2)$

Consider the following pseudocode:

```
Algorithm IterativeFunction (a)  // a is a non-negative integer
  b ← 0
  while (a > 0)
    b ← 10 * b + (a % 2)
    a ← a / 2
  endwhile
  return b
```

15. (2 points) What is the time complexity of the `IterativeFunction` pseudocode shown above?
    A. $\Theta(a)$
    B. $\Theta(\log a)$
    C. $\Theta(1)$
    D. $\Theta(a/2)$
    E. $\Theta(a\%2)$

16. (2 points) What is the space complexity of the `IterativeFunction` pseudocode shown above?
    A. $\Theta(a)$
    B. $\Theta(\log a)$
    C. $\Theta(1)$
    D. $\Theta(a/2)$
    E. $\Theta(a\%2)$

Consider the following pseudocode:

```
Algorithm RecursiveFunction2 (A)  // A is an array of n items (1 or more)
  if (A.size = 1)
    return A[0]
  else
    B ← A[1 : A.size-1]  // creates new array B of n-1 items
    return max(A[0], RecursiveFunction2 (B))
  endif
```

17. (2 points) What is the time complexity of the `RecursiveFunction2` pseudocode shown above?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

18. (2 points) What is the space complexity of the `RecursiveFunction2` 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 IterativeFunction2 (A)  // A is an array of n items (1 or more)
  B ← A  // creates new array B of n items
  m ← B[0]
  while (B.size > 1)
    B ← B[1 : B.size-1]  // replaces B with array of one fewer items
    m ← max(m, B[0])
  endwhile
  return m
```

19. (2 points) What is the time complexity of the `IterativeFunction2` pseudocode shown above?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

20. (2 points) What is the space complexity of the `IterativeFunction2` 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 IV. Particular Sorting/Searching Algorithms

Note: For questions in this section, be sure to consider that there may be best, average, and worst cases for each algorithm. If a question does not indicate which case to consider, you should consider the case most appropriate for answering the given question. Also, these questions refer to time complexity, not space complexity.

21. (2 points) Which of the following approaches are most appropriate for searching for a given value in an *unsorted* list?
    - ***A. Linear search***
    - B. Binary search
    - C. Random search
    - D. A and B
    - E. B and C

22. (2 points) Which of the following approaches are appropriate for searching for a given value in a *sorted* list?
    - A. Linear search
    - ***B. Binary search***
    - C. Random search
    - D. A and B
    - E. B and C

23. (2 points) Which of the following are true?
    - ***A. Straight Insertion Sort*** $\in O(n^2)$
    - B. Straight Insertion Sort $\in O(n \log n)$
    - C. Straight Insertion Sort $\in O(n)$
    - D. B and C
    - E. None of the above

24. (2 points) Which of the following are true?
    - A. Straight Insertion Sort $\in \Omega(n^2)$
    - B. Straight Insertion Sort $\in \Omega(n \log n)$
    - ***C. Straight Insertion Sort*** $\in \Omega(n)$
    - D. B and C
    - E. None of the above

25. (2 points) Which of the following are true?
    - A. Straight Insertion Sort $\in \Theta(n^2)$
    - B. Straight Insertion Sort $\in \Theta(n \log n)$
    - C. Straight Insertion Sort $\in \Theta(n)$
    - D. B and C
    - ***E. None of the above***

26. (2 points) Which of the following are true?
    - ***A. Bubble Sort*** $\in O(n^2)$
    - B. Bubble Sort $\in O(n \log n)$
    - C. Bubble Sort $\in O(n)$
    - D. All of the above
    - E. None of the above

27. (2 points) Which of the following are true?
    - A. Bubble Sort $\in \Omega(n^2)$
    - B. Bubble Sort $\in \Omega(n \log n)$
    - C. Bubble Sort $\in \Omega(n)$
    - ***D. All of the above***
    - E. None of the above

28. (2 points) Which of the following are true?
    - ***A. Bubble Sort*** $\in \Theta(n^2)$
    - B. Bubble Sort $\in \Theta(n \log n)$
    - C. Bubble Sort $\in \Theta(n)$
    - D. All of the above
    - E. None of the above

29. (2 points) Which of the following are true?
    - ***A. Quick Sort*** $\in O(n^2)$
    - B. Quick Sort $\in O(n \log n)$
    - C. Quick Sort $\in O(n)$
    - D. B and C
    - E. None of the above

30. (2 points) Which of the following are true?
    - A. Quick Sort $\in \Omega(n^2)$
    - B. Quick Sort $\in \Omega(n \log n)$
    - C. Quick Sort $\in \Omega(n)$
    - ***D. B and C***
    - E. None of the above

31. (2 points) Which of the following are true?
    - A. Quick Sort $\in \Theta(n^2)$
    - B. Quick Sort $\in \Theta(n \log n)$
    - C. Quick Sort $\in \Theta(n)$
    - D. B and C
    - ***E. None of the above***

32. (2 points) Which of the following algorithms would be best to sort a randomly ordered array of data?
    - A. Straight Insertion Sort
    - B. Bubble Sort
    - ***C. Quick Sort***
    - D. All of the above
    - E. None of the above

33. (2 points) Which of the following algorithms would be best to sort a nearly sorted array of data?
    - ***A. Straight Insertion Sort***
    - B. Bubble Sort
    - C. Quick Sort
    - D. All of the above
    - E. None of the above

34. (2 points) Which of the following is an advantage of Quick Sort over Merge Sort?
    - A. Better average runtime equivalence class
    - B. Stability with respect to previously sorted results
    - ***C. Sorts within existing array***
    - D. All of the above
    - E. None of the above

35. (2 points) Which of the following is an advantage of Merge Sort over Quick Sort?
    - A. Better average runtime equivalence class
    - ***B. Stability with respect to previously sorted results***
    - C. Sorts within existing array
    - D. All of the above
    - E. None of the above

Short Answer Question 1: 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	26	40	48	61	86	94	57	16	25	11
index	0	1	2	3	4	5	6	7	8	9

Additional space to answer Short Answer Question 1