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.

Part I. Data Structures Concepts

1. (2 points) Which data structure exhibits a first in, first out (FIFO) policy?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

2. (2 points) Which data structure exhibits a first in, last out (FILO) policy?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

3. (2 points) Which data structure allows for efficient lookups based on keys?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

4. (2 points) Which data structure allows for efficient lookups based on index (position)?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

5. (2 points) Which data structure allows for efficient sorting?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

6. (2 points) Which data structure allows for efficient merging?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

Part II. Time Complexity of Linked Lists

7. (2 points) What is the time complexity for efficiently *inserting* an arbitrary item into an *unsorted* linked list?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(n \log_2 n)$

8. (2 points) What is the time complexity for efficiently *finding* a particular item in an *unsorted* linked list?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(n \log_2 n)$

9. (2 points) What is the time complexity for efficiently *inserting* an arbitrary item into a *sorted* linked list?
   A. $\Theta(n)$
   B. $\Theta(\log_2 n)$
   C. $\Theta(1)$
   D. $\Theta(n^2)$
   E. $\Theta(n \log_2 n)$

10. (2 points) What is the time complexity for efficiently *finding* a particular item in a *sorted* linked list?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

11. (2 points) What is the time complexity for efficiently *removing* an item from a singly linked list, once it has been found?
    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 time complexity for efficiently *sorting* a linked list?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

Part III. Time Complexity of Stacks and Queues

13. (2 points) What is the time complexity for efficiently *adding* an item to a *stack* that is implemented as a linked list?
    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 time complexity for efficiently *removing* an item from a *stack* that is implemented as a linked list?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

15. (2 points) What is the time complexity for efficiently *adding* an item to a *queue* that is implemented as a linked list?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

16. (2 points) What is the time complexity for efficiently *removing* an item from a *queue* that is implemented as a linked list?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

17. (2 points) What is the time complexity for efficiently *adding* an item to a *priority queue* that is implemented as a single linked list?
    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 time complexity for efficiently *adding* an item to a *priority queue* that is implemented as an array of linked lists (one linked list per priority level)?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

Part IV. Time Complexity of Hash Tables

19. (2 points) What is the time complexity of adding an item to a hash table that uses a *perfect* hash function as the table becomes full?
    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 time complexity of adding an item to a hash table that uses an *imperfect* hash function as the table becomes full?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

21. (2 points) What is the time complexity of efficiently resizing a hash table?
    A. $\Theta(n)$
    B. $\Theta(\log_2 n)$
    C. $\Theta(1)$
    D. $\Theta(n^2)$
    E. $\Theta(n \log_2 n)$

Part V. Hash Table Variations

22. (2 points) Which is a drawback of using modulus arithmetic as a hash function for a table with integer keys?
    A. Modulus arithmetic doesn't apply to integers
    B. The hash values are likely to cluster
    C. Modulus arithmetic is slow
    D. Hash values must fall within table limits
    E. Characters can be converted to integers using their ASCII values

23. (2 points) Which is a drawback of using linear probing as a collision resolution technique?
    A. Linear probing is expensive to compute
    B. Hash values must fall within table limits
    C. Hash value clusters cause repeated probes
    D. Secondary clustering is as much of a problem as primary clustering
    E. Secondary clustering occurs with quadratic probing

24. (2 points) A hashing function is considered perfect if which is true?
    A. The function can be computed in constant time
    B. All hash values computed are within table limits
    C. All keys hash to unique values
    D. All hash values computed are mersenne primes
    E. Modulus arithmetic is not required

25. (2 points) Which is true of linear hashing?
    A. Each overflow is placed in the next available bucket
    B. A bucket is split each time there is an insertion
    C. It uses quadratic probing
    D. It uses double hashing
    E. Buckets are split in a predefined order

Exam continues with short answer questions.

Short Answer Question 1: Radix Sort (10 points)

```
// Radixsort takes:
// A: the array to sort
// r: the radix (base) for the keys to be sorted
// d: the number of digits (of the given radix) in each key
Algorithm Radixsort (A, r, d)
  create Q[r] // Q is an array of r queues, all initially empty
  for k from 0 to d-1
    for i from 0 to A.size
      Q[(A[i].key/(r to the power k)) modulus r].enqueue(A[i])
    end for i
    i ← 0
    for j from 0 to r do
      while Q[j] is not empty
        A[i] ← Q[j].dequeue()
        i ← i + 1
      end while
    end for j
  end for k
```

Given `r` is 10 and `d` is 2, show the steps followed by the Radixsort algorithm given above in pseudocode when sorting the following array. Draw one figure for `Q` and one figure for `A` for each value of `k`.

value	88	06	63	36	43	16	11	32	13	25
index	0	1	2	3	4	5	6	7	8	9

Additional space to answer Short Answer Question 1.

Short Answer Question 2: Time Complexity of Sorting (10 points)
According to Eppstein, 1996 (`https://www.ics.uci.edu/~eppstein/161/960123.html`):

> "Is [radix sort with base 10] ever the best algorithm to use?
>
> Answer: No. If k is smaller than n, this takes O(n log k) while bucket sort takes only O(n). And if k is larger than n, the O(n log k) taken by this method is worse than the O(n log n) taken by comparison sorting."

In this quote, k is the maximum value of a key (and the minimum value is 0) while n is the number of items to sort.

Explain one way in which the quoted claim is wrong.

Short Answer Question 3: Data Structures Comparison (10 points)

Explain why the time complexity of inserting an arbitrary item into a unsorted linked list is sometimes, but not always, better than the time complexity of inserting an arbitrary item into an unsorted vector.

Short Answer Question 4: Hashing (10 points)

A. Given the following items to insert into a hash table of size 10, **show the hash table after all items have been inserted**.

- The items are to be inserted starting from the top of the list and working down.
- The hash function is key modulus table size.
- The collision resolution strategy is linear probing.

Items to Insert

Item	Key
A	435
B	485
C	625
D	454
E	178
F	846
G	871
H	187
I	316

Hash Table

Bucket	Item
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

B. Calculate the load factor after all of the items have been inserted. **Show your work.**

C. Calculate the average number of probes needed for successful search over all of the keys in the original item set, after all of the items have been inserted. **Show your work.**

Short Answer Question 5: Linear Hashing (10 points)

A. Given the following items to insert into a hash table that uses linear hashing, **show the hash table *during and* after all items have been inserted**. (That is, rather than erasing when things change, cross them out as they change so that I can still see them.)

- The items are to be inserted starting from the top of the list and working down.
- The collision resolution strategy is separate chaining.

Items to Insert

Item	Key
A	1011
B	0111
C	1001
D	1010
E	1111
F	0110
G	1100
H	0001
I	0010

Hash Table

Bucket	Item

B. Calculate the load factor after all of the items have been inserted. **Show your work.**

C. Calculate the average number of probes needed for successful search over all of the keys in the original item set, after all of the items have been inserted. **Show your work.**