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 always adds and removes elements from the same end?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

2. (2 points) Which data structure always adds and removes elements from opposite ends?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

3. (2 points) Which data structure adds elements in the middle by moving one or more other element(s) out of the way?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

4. (2 points) Which data structure adds elements in the middle without needing to move any of the elements already present?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

5. (2 points) Which data structure tries to add each element at a preferred location first but will add it elsewhere if the preferred location is already occupied?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

6. (2 points) Which data structure grows with every element added?
   A. Stack
   B. Queue
   C. Hash Table
   D. Array
   E. Linked List

Part II. Using Data Structures

7. (2 points) Which method should be used to find an arbitrary item in an *unsorted linked list*?
   A. Linear search
   B. Binary search
   C. Peek
   D. Pop
   E. Dequeue

8. (2 points) Which method should be used to find an arbitrary item in a *sorted linked list*?
   A. Linear search
   B. Binary search
   C. Peek
   D. Pop
   E. Dequeue

9. (2 points) Which method should be used to view the front item in a *queue*?
   A. Linear search
   B. Binary search
   C. Peek
   D. Pop
   E. Dequeue

10. (2 points) Which method should be used to view the top item in a *stack*?
    A. Linear search
    B. Binary search
    C. Peek
    D. Pop
    E. Dequeue

11. (2 points) Which method should be used to *remove* the front item from a *queue*?
    A. Linear search
    B. Binary search
    C. Peek
    D. Pop
    E. Dequeue

12. (2 points) Which method should be used to *remove* the top item from a *stack*?
    A. Linear search
    B. Binary search
    C. Peek
    D. Pop
    E. Dequeue

Part III. Implementations

13. (2 points) Which data structure(s) should be used to implement a *stack* to allow it to grow even if contiguous memory is not available?
    A. Array
    B. In-place Resizable Array
    C. Linked List
    D. A and C
    E. B and C

14. (2 points) Which data structure(s) should be used to implement a *queue* to allow it to grow even if contiguous memory is not available?
    A. Array
    B. In-place Resizable Array
    C. Linked List
    D. A and C
    E. B and C

15. (2 points) Which data structure(s) should be used to implement *linear hashing with separate chaining*?
    A. Array
    B. In-Place Resizable Array
    C. Linked List
    D. A and C
    E. B and C

16. (2 points) Which data structure(s) should be used to implement *a circular queue*?
    A. Array
    B. In-Place Resizable Array
    C. Linked List
    D. A and C
    E. B and C

17. (2 points) Which data structure(s) should be used to implement *radix sort*?
    A. Queue
    B. Stack
    C. Array
    D. A and C
    E. B and C

Part IV. Comparisons

18. (2 points) Which data structure is generally the most efficient for retrieval of a data item by key?
    A. Unsorted Linked List
    B. Unsorted Resizable Array
    C. Sorted Resizable Array
    D. Hash Table
    E. They're all equal in this respect

19. (2 points) Which data structure is generally the *second* most efficient for retrieval of a data item by key?
    A. Unsorted Linked List
    B. Unsorted Resizable Array
    C. Sorted Resizable Array
    D. Hash Table
    E. They're all equal in this respect

20. (2 points) Which data structure is the most consistent for insertion time?
    A. Unsorted Linked List
    B. Unsorted Resizable Array
    C. Resizable Resizable Array
    D. Hash Table
    E. They're all equal in this respect

21. (2 points) Which algorithm has a maximum Big Ω bound lower than $\Omega(n \log n)$ because it does not do a comparison-based sort?
    A. Hashing n items
    B. Merge Sort on n items
    C. Radix Sort on n items
    D. Quick Sort on n items
    E. Shell Sort on n items

Part V. Hash Tables

22. (2 points) How are hash functions used in hash tables?
    A. To map from keys to table indices
    B. To generate random numbers
    C. To maintain insertion order
    D. To dynamically grow the table
    E. To create tags that are searchable

23. (2 points) Which collision resolution strategy is most likely to result in primary clustering?
    A. Linear probing
    B. Quadratic probing
    C. Double hashing
    D. Separate chaining
    E. Circular queues

24. (2 points) Which collision resolution strategy is most likely to result in secondary clustering?
    A. Linear probing
    B. Quadratic probing
    C. Double hashing
    D. Separate chaining
    E. Circular queues

25. (2 points) A perfect hash function avoids which of the following problems?
    - A. Collisions
    - B. Primary clustering
    - C. Secondary clustering
    - D. Declining performance with increasing load factor
    - E. All of the above

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	48	37	24	19	81	83	84	51	50	53
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 Hashing (20 points)

A. List and **explain** the maximum Big Ω time complexity for finding an arbitrary item in an arbitrary hash table.

B. Compare the answer you gave in Part A to the maximum Big Ω time complexity for finding an arbitrary item in an arbitrary sorted array. **Explain** which (if either) is better and how that performance is achieved.

Short Answer Question 3: 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 primary hash function is key modulus table size.
- The collision resolution strategy is double hashing.
- The secondary hash function is key div table size, where "div" is integer division (that is, division discarding the remainder).

Items to Insert

Item	Key
A	35
B	49
C	34
D	93
E	84
F	96
G	11
H	22
I	38

Hash Table

Bucket Number	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 4: 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	0110
B	0111
C	0011
D	1001
E	1010
F	1011
G	1110
H	1111
I	1101

Hash Table

Bucket Number	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.**