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. Basic Data Structures

1. (2 points) Which data structure requires a contiguous block of memory?
   - A. Array
   - B. Resizable Array
   - C. Linked List
   - ***D. A and B***
   - E. All of the above

2. (2 points) Which data structure tends to cause memory fragmentation?
   - A. Array
   - ***B. Resizable Array***
   - C. Linked List
   - D. A and B
   - E. All of the above

3. (2 points) Which data structure can be defined recursively?
   - A. Array
   - B. Resizable Array
   - ***C. Linked List***
   - D. A and B
   - E. All of the above

4. (2 points) Which data structure can be used to implement a queue?
   - A. Array
   - B. Resizable Array
   - C. Linked List
   - D. A and B
   - ***E. All of the above***

5. (2 points) Which data structure can be used to implement a stack?
   - A. Array
   - B. Resizable Array
   - C. Linked List
   - D. A and B
   - ***E. All of the above***

6. (2 points) Which data structure can be used to implement a hash table?
   - A. Array
   - B. Resizable Array
   - C. Linked List
   - ***D. A and B***
   - ***E. All of the above***

Part II. Using Data Structures

7. (2 points) Which data structure is most appropriate for efficiently searching a sorted list?
   - ***A. Array***
   - B. Linked List
   - C. Stack
   - D. Queue
   - E. Hash Table

8. (2 points) Which data structure is most appropriate for round-robin scheduling (in which the scheduler cycles repeatedly through all ready processes)?
   - A. Array
   - B. Linked List
   - C. Stack
   - ***D. Queue***
   - E. Hash Table

9. (2 points) Which data structure is most appropriate for temporarily storing function, procedure, and/or method call information such as parameters, local variables, and program counters?
   - A. Array
   - B. Linked List
   - ***C. Stack***
   - D. Queue
   - E. Hash Table

10. (2 points) Which data structure is most appropriate when quickly finding an arbitrary element is more important than space efficiency?
    - A. Array
    - B. Linked List
    - C. Stack
    - D. Queue
    - ***E. Hash Table***

11. (2 points) Which data structure is most appropriate when incrementally allocating only the space necessary to store incoming data is more important than quickly finding an arbitrary element?
    - A. Array
    - ***B. Linked List***
    - C. Stack
    - D. Queue
    - E. Hash Table

12. (2 points) Which data structure is most appropriate for sorting a list of data?
    - ***A. Array***
    - B. Linked List
    - C. Stack
    - D. Queue
    - E. Hash Table

Part III. Data Structures Methods

13. (2 points) Which method is most appropriate for adding an element to a **hash table**?
    - A. `insert(element, index)`
    - B. `enqueue(element)`
    - C. `push(element)`
    - [✓ ***D.*** `add(element, key)`]
    - E. `peek()`

14. (2 points) Which method is most appropriate for adding an element to an **array**?
    - [✓ ***A.*** `insert(element, index)`]
    - B. `enqueue(element)`
    - C. `push(element)`
    - D. `add(element, key)`
    - E. `peek()`

15. (2 points) Which method is most appropriate for adding an element to a **queue**?
    - A. `insert(element, index)`
    - [✓ ***B.*** `enqueue(element)`]
    - C. `push(element)`
    - D. `add(element, key)`
    - E. `peek()`

16. (2 points) Which method is most appropriate for adding an element to a **stack**?
    - A. `insert(element, index)`
    - B. `enqueue(element)`
    - [✓ ***C.*** `push(element)`]
    - D. `add(element, key)`
    - E. `peek()`

17. (2 points) Which method is most appropriate for adding an element to a **linked list**?
    - [✓ ***A.*** `insert(element, index)`]
    - B. `enqueue(element)`
    - C. `push(element)`
    - D. `add(element, key)`
    - E. `peek()`

Part IV. Time and Space Complexity

18. (2 points) Which of the following operations has a time complexity of $\Theta(1)$?
    - A. Calculating a hash value
    - B. Indexing into a hash table based on hash value
    - C. Using a collision resolution strategy
    - [✓ ***D. A and B***]
    - E. All of the above

19. (2 points) Which of the following operations has a time complexity of $\Theta(1)$?
    - A. Inserting an arbitrary element into an unsorted linked list
    - B. Removing an arbitrary element from an unsorted linked list (once it has been found)
    - C. Finding an arbitrary element in an unsorted linked list
    - ***D. A and B***
    - E. All of the above

20. (2 points) Which of the following operations has a time complexity of $\Theta(1)$?
    - A. Adding an item to a stack
    - B. Removing an item from a stack
    - C. Looking at the top item on a stack
    - D. A and B
    - ***E. All of the above***

21. (2 points) Which of the following operations has a time complexity of $\Theta(1)$?
    - A. Adding an item to a queue
    - B. Removing an item from a queue
    - C. Looking at the first item in a queue
    - D. A and B
    - ***E. All of the above***

Part V. Hash Tables

22. (2 points) To ensure that all operations on a hash table operate in constant time, we need which of the following?
    - ***A. A perfect hash function***
    - B. A minimal hash function
    - C. An ideal collision resolution strategy
    - D. A and B
    - E. All of the above

23. (2 points) The collision resolution strategy of looking at the next bucket to see if it is available is which of the following?
    - ***A. Linear probing***
    - B. Quadratic probing
    - C. Double hashing
    - D. Separate chaining
    - E. None of the above

24. (2 points) The collision resolution strategy of looking at buckets based on an additional function that translates keys to values is which of the following?
    - A. Linear probing
    - B. Quadratic probing
    - ***C. Double hashing***
    - D. Separate chaining
    - E. None of the above

25. (2 points) The collision resolution strategy of using auxiliary linked lists for bucket overflows is which of the following?
    - A. Linear probing
    - B. Quadratic probing
    - C. Double hashing
    - *D. Separate chaining*
    - E. None 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	6	13	94	32	60	9	7	70	93	49
index	0	1	2	3	4	5	6	7	8	9

Additional space to answer Short Answer Question 1.

Short Answer Question 2: 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	69
B	66
C	80
D	35
E	18
F	60
G	89
H	70
I	12

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 3: Double Hashing (10 points)

Describe and **explain** a shortcoming of the secondary hash function given in Short Answer Question 2.

Short Answer Question 4: 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 a hash table.

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	1010
B	1001
C	1100
D	1111
E	0011
F	1101
G	0010
H	0001
I	0110

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.**