# Western Governors University (WGU) ICSC2100 C949 Data Structures and Algorithms I Practice Exam (Sample)

## Study Guide

BY EXAMZIFY

**Everything you need from our exam experts!**

**Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.**

# Table of Contents

# Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn’t just about memorization, it’s about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you’re preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You’ll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it’s correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It’s ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn’t about getting every question right the first time, it’s about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let’s begin.

# How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here’s how to get the most out of your Examzify study guide:

### 1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

### 2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

### 3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

### 4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

### 5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

### 6. Repeat and Review

Don’t just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There’s no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

# Questions

1. **What is the equation for finding the middle value of an array?**

   A. (high + low)/2

   B. (low + high)/2

   C. (min + max)/2

   D. (start + end)/2

2. **Which of the following sorting algorithms has the best average time complexity?**

   A. Bubble sort

   B. Merge sort

   C. Quicksort

   D. Both Merge sort and Quicksort

3. **What is typically the initial state of a newly created doubly linked list?**

   A. Empty with no nodes

   B. Contains one node only

   C. Contains two nodes

   D. Filled with dummy nodes

4. **What is the result of the hash function key % 10 for a given key?**

   A. Indices from 1 to 10

   B. Indices from 0 to 10

   C. Indices from 0 to 9

   D. Indices from 0 to 100

5. **Which of the following describes a First-in, First-out (FIFO) data container?**

   A. Stack

   B. Queue

   C. Tree

   D. Doubly Ended Queue

**6. What is the term for an integer counter that represents how many variables reference an object?**

**A. Reference Count**

**B. Object Count**

**C. Variable Count**

**D. Pointer Count**

**7. Which data structure is characterized by its capability to branch into child nodes?**

**A. Graph**

**B. Array**

**C. Tree**

**D. Stack**

**8. In data structures, what is a common characteristic of a doubly linked list?**

**A. Nodes have no references**

**B. Nodes can only be accessed linearly**

**C. Nodes reference both the next and previous nodes**

**D. No node references**

**9. When an item is removed from a queue, it comes from which part?**

**A. Middle**

**B. Head**

**C. Tail**

**D. Bottom**

**10. Which sorting algorithm is known for selecting a 'pivot' element?**

**A. Bubble sort**

**B. Merge sort**

**C. Quicksort**

**D. Insertion sort**

# Answers

**1. A**
**2. D**
**3. A**
**4. C**
**5. B**
**6. A**
**7. C**
**8. C**
**9. B**
**10. C**

# Explanations

## 1. What is the equation for finding the middle value of an array?

A. (high + low)/2

B. (low + high)/2

C. (min + max)/2

D. (start + end)/2

**The equation for finding the middle value of an array involves determining the average of the high and low indices that define the current range of the array being considered. Using the formula (high + low) / 2 effectively gives you the midpoint between these two indices. This is particularly useful in algorithms such as binary search, where you need to divide the array into two halves to efficiently locate a specific element. By calculating the average of the high and low indices, you determine where the midpoint lies, allowing you to narrow down your search space. While other options mention concepts of range or midpoint, they may not accurately reflect the correct method for finding the middle index specifically in the context of an array where the indices are defined as high and low. Hence, the formula (high + low) / 2 adequately captures the essence of finding the median or middle index for arrays and serves as a fundamental operation in algorithm design.**

## 2. Which of the following sorting algorithms has the best average time complexity?

A. Bubble sort

B. Merge sort

C. Quicksort

D. Both Merge sort and Quicksort

**The best average time complexity among the provided sorting algorithms is exhibited by both Merge sort and Quicksort, making the option that includes both of these algorithms the most accurate choice. Merge sort has a guaranteed average time complexity of O(n log n). This efficiency comes from its divide-and-conquer approach, where the array is recursively divided into halves until individual elements are reached, and then these elements are merged back together in sorted order. This systematic splitting and merging process maintains a consistent performance regardless of the nature of the input data. Quicksort also boasts an average time complexity of O(n log n), which similarly arises from its divide-and-conquer strategy. It selects a 'pivot' element, partitions the array into elements less than and greater than the pivot, and then recursively sorts the subarrays. Despite its average-case efficiency, quicksort can degrade to O(n^2) in the worst-case scenario (for instance, if the smallest or largest element is consistently chosen as the pivot), but this is typically avoided with good pivot selection strategies. Given that both Merge sort and Quicksort achieve the same time complexity on average, this makes the combination of the two the best option among the provided choices. The other sorting algorithms, such as Bubble sort**

## 3. What is typically the initial state of a newly created doubly linked list?

A. Empty with no nodes

B. Contains one node only

C. Contains two nodes

D. Filled with dummy nodes

**A newly created doubly linked list typically starts in an empty state, meaning it has no nodes at all. In this initial state, both the "head" (the starting point of the list) and the "tail" (the endpoint of the list) are set to null or None, indicating that there are no nodes present and thus, no data to traverse or manipulate. This behavior is consistent with the general approach to data structure initialization, where a structure begins without elements until explicitly populated.  When implementing a doubly linked list, operations such as insertion, deletion, and traversal depend on this initial empty state. Once nodes are added, the list will transition from this state to having one or more nodes, which the implementation supports through appropriate methods that update the head, tail, and links between nodes accordingly.**

## 4. What is the result of the hash function key % 10 for a given key?

A. Indices from 1 to 10

B. Indices from 0 to 10

C. Indices from 0 to 9

D. Indices from 0 to 100

**The result of the hash function key % 10 for a given key produces values within a specific range based on modulo arithmetic. When you calculate key % 10, the output will be the remainder when the key is divided by 10. This remainder can only be one of the values from 0 up to, but not including, 10.   Since the possible remainders for this operation are 0 through 9, it means that the function effectively maps any integer key to an index within this range. Thus, the output of the hash function will create indices that correspond to the values from 0 to 9, making this the correct interpretation of the result of the given hash function.   Therefore, the correct answer reflects that when working with a hash function defined in this way, it systematically compresses a potentially infinite range of input keys into a fixed-size array or structure with indices limited to this range.**

### 5. Which of the following describes a First-in, First-out (FIFO) data container?

A. Stack

B. Queue

C. Tree

D. Doubly Ended Queue

**A First-in, First-out (FIFO) data container is characterized by the principle that the first element added to the container will be the first one to be removed. This mirrors the way that a line of people operate: the person who arrives first is the one who gets served first. In the context of this question, a queue embodies the FIFO principle, which means that when elements are inserted, they go to the back of the queue, and when elements are removed, they come off the front. This allows for orderly processing and ensures that tasks are completed in the order they were received.  On the other hand, a stack, which is based on a Last-in, First-out (LIFO) structure, allows the most recently added element to be the first one removed. Trees are hierarchical structures used to model relationships, and a doubly-ended queue (deque) does allow for insertion and removal from both the front and the back, but it does not confine itself strictly to FIFO behavior.   Thus, the queue is the correct representation of a FIFO data structure, as it maintains the order of elements in exact accordance with their arrival sequence.**

### 6. What is the term for an integer counter that represents how many variables reference an object?

A. Reference Count

B. Object Count

C. Variable Count

D. Pointer Count

**The term that describes an integer counter reflecting the number of variables referencing an object is known as the reference count. This concept is integral in memory management, particularly in garbage collection systems, where it helps in determining when an object in memory can be safely deallocated. When the reference count of an object drops to zero, it indicates that no variables are referencing that object anymore, allowing the system to reclaim that memory.  In contrast, the other terms listed do not accurately define this concept. "Object Count" could imply the total number of objects in memory, but it does not specifically relate to the references to a particular object. "Variable Count" might suggest a number of variable instances, yet it doesn't convey the notion of how many of those link to a given object. "Pointer Count" similarly suggests a count of pointers but lacks the specificity of reference counting as it pertains to actual references to an object rather than just pointers in memory. Thus, the most precise term for describing this scenario is indeed the reference count.**

## 7. Which data structure is characterized by its capability to branch into child nodes?

A. Graph

B. Array

C. Tree

D. Stack

**The correct option is based on the defining characteristics of a tree data structure. A tree is a hierarchical structure that consists of nodes connected by edges, and it is designed to branch into child nodes. Each tree has a root node, which serves as the starting point, and all other nodes are either its descendants or leaf nodes which do not have any children. The branching capability of trees allows them to represent relationships and hierarchies effectively, making it ideal for various applications such as representing file systems, organizational structures, or developing decision trees. Each node in a tree can have multiple children, which differentiates it from linear structures, where elements are arranged in a single line, such as in arrays or stacks. Trees provide efficient ways of storing and manipulating data that rely on hierarchical relationships, facilitating operations such as searching, insertion, and deletion in a way that can be more efficient than other data structures.**

## 8. In data structures, what is a common characteristic of a doubly linked list?

A. Nodes have no references

B. Nodes can only be accessed linearly

C. Nodes reference both the next and previous nodes

D. No node references

**A doubly linked list is characterized by each of its nodes having two references: one pointing to the next node in the sequence and another pointing to the previous node. This structure allows for traversal in both directions—forward and backward—thereby providing greater flexibility compared to a singly linked list, which only allows traversal in one direction. The ability to access nodes from both ends is particularly useful for certain algorithms and operations, such as inserting or deleting nodes, as it simplifies these processes by allowing links to be navigated both ways. This feature enhances performance for various tasks, making the doubly linked list a valuable data structure in programming and algorithm design.**

## 9. When an item is removed from a queue, it comes from which part?

A. Middle

B. Head

C. Tail

D. Bottom

In a queue, items are managed in a First-In-First-Out (FIFO) manner, meaning that the first item added to the queue is the first one to be removed. This structure is akin to a line of people waiting for service, where the individual at the front of the line is served first. When an item is removed from a queue, it is taken from the front of the queue, often referred to as the head. This design ensures that new items can be added at the rear or tail of the queue while maintaining the order of processing for those already in the queue. In practical terms, if you visualize the queue as a series of elements lined up, you will find that removing an element corresponds specifically to the head of this arrangement. This reinforces the queue's characteristic of maintaining order based on the sequence of arrival, making the head the logical point to retrieve items from as they are processed.

## 10. Which sorting algorithm is known for selecting a 'pivot' element?

A. Bubble sort

B. Merge sort

C. Quicksort

D. Insertion sort

Quicksort is a highly efficient sorting algorithm that utilizes the concept of a 'pivot' element as a key aspect of its sorting process. In this algorithm, the pivot is chosen from the array elements, and the other elements are rearranged into two partitions: those less than the pivot and those greater than the pivot. This process is recursive, meaning that quicksort will repeatedly select new pivots and partition the subarrays until the entire array is sorted. The selection of a pivot is crucial as it significantly impacts the performance of the algorithm; ideally, a good pivot will lead to evenly sized partitions, resulting in optimal efficiency. Quicksort has an average time complexity of O(n log n), making it faster than various other sorting algorithms for large datasets in most practical applications.

# Next Steps

**Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.**

**As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.**

**If you need help, have suggestions, or want to share feedback, we’d love to hear from you. Reach out to our team at hello@examzify.com.**

**Or visit your dedicated course page for more study tools and resources:**

**https://wgu-icsc2100-c949-datastructuresandalgorithmsi.examzify.com**

**We wish you the very best on your exam journey. You've got this!**