

CS 14: Introduction to
Data Structures & Algorithms

April 22, 2003
Quiz 1, Form:

First Name: _____

Last Name: _____

ID Number: _____

Signature: _____

This test contains two sections, a **MULTIPLE-CHOICE** section and a **SHORT-ANSWER** section. Please make sure to pace yourself properly so that you have time to work on both sections.

You may use **ONLY** the test's last sheet (front and back pages) for **scratch** work; I will **not** look at your scratch work and **nothing** there will be graded. **Do NOT** remove the scratch pages.

Good luck. :)

Part 1. MULTIPLE-CHOICE section. Please choose **only one** of the alternatives provided for each question and remember to mark your answers on the scantron form because **ONLY THOSE** will be considered. If scantron forms are not being used in this test, then make sure to **CIRCLE** your chosen answer.

Each question is worth **1 point** and **NO partial credit** is given. No negative points are given for wrong answers. Please note that you will **NOT** get any credit for leaving questions unanswered.

1. The operation enqueue() affects
 - (a) the top of the queue.
 - (b) the front of the queue.
 - (c) the back of the queue.
 - (d) both the front and the back of the queue.
 - (e) either the front or the back of the queue, but not both at the same time, depending on the user's needs.
2. **FIFO** describes the behavior of a **queue**.
 - (a) True.
 - (b) False.

3. Consider the following code fragment, where `lptr` is a pointer to an instance of a class implementing the **ADT List**. Assume that the list elements are indexed in the same way an array is.

```
while (( lptr -> getLength() ) > 0)
    lptr -> deleteAtIndex (0);
```

The code fragment above:

- (a) deletes the first element of the list.
 - (b) leaves the list empty.
 - (c) goes into an infinite loop if the list has more than one element.
 - (d) has a **worst-case** runtime complexity of $\mathcal{O}[n]$.
 - (e) None of the above.
4. Consider the following code fragment, where `lptr` is a pointer to an instance of a class implementing the **ADT List**. Assume a pointer implementation with no **tail** pointer, but with a **length** member variable. Also assume that the list elements are indexed in the same way an array is.

```
while (( lptr -> getLength() ) > 0)
    lptr -> deleteAtIndex ( ( lptr -> getLength() ) - 1 );
```

The code fragment above has a **worst-case** runtime complexity of:

- (a) $\mathcal{O}[\log_2(n)]$.
 - (b) $\mathcal{O}[n]$.
 - (c) $\mathcal{O}[n^2]$.
 - (d) $\mathcal{O}[n^3]$.
 - (e) None of the above.
5. The **worst-case** runtime complexity of **linear search** when applied to a problem of size n is
- (a) Logarithmic.
 - (b) $\mathcal{O}[n]$.
 - (c) $\mathcal{O}[n \log_2(n)]$.
 - (d) $\mathcal{O}[n^2]$.
 - (e) Constant.

6. **Black-box** testing of an ADT is a testing procedure in which you don't have access to the ADT's
- source code.
 - interface.
 - axioms.
 - All of the above.
 - None of the above.
7. What is the correct **increasing** ordering of the following runtime complexities: $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^n]$, $\mathcal{O}[n]$, $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^2]$? Recall that $n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$.
- $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n]$, $\mathcal{O}[n^2]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^n]$.
 - $\mathcal{O}[n^n]$, $\mathcal{O}[n]$, $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^2]$.
 - $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^n]$, $\mathcal{O}[n]$, $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^2]$.
 - $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^2]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^n]$.
 - $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^2]$, $\mathcal{O}[n^n]$, $\mathcal{O}[n!]$.
8. Deleting the **last** element of a **singly**-linked list with a **tail** member variable has a **worst-case** runtime complexity of
- $\mathcal{O}[n \log_2(n)]$.
 - $\mathcal{O}[n]$.
 - $\mathcal{O}[n^2]$.
 - $\mathcal{O}[\log_2(n)]$.
 - $\mathcal{O}[1]$.
9. A **stack** is implemented using a statically defined array of size 100. After the stack is instantiated, 31 pushes and 17 pops are performed on it (not necessarily all pushes before all the pops — assume that they could have been done in any order). Which of the statements below is correct after these operations are performed?
- The stack has 100 elements.
 - The stack has 14 elements.
 - The order in which the operations are performed is important.
 - Both (a) and (c).
 - Both (b) and (c).

10. An algorithm with **worst-case** runtime complexity proportional to $\log_2(n)$ _____ than another algorithm with **worst-case** runtime complexity proportional to n .
- (a) is always faster
 - (b) is always slower
 - (c) may be faster
 - (d) may be slower
11. A **List Iterator**
- (a) is an object with responsibilities similar to that of an array index.
 - (b) lets you move from node to node without requiring you to know the details of how the list or the node structures are implemented.
 - (c) is also an ADT.
 - (d) Both (a) and (b).
 - (e) All of the above.
12. The **binary search** algorithm is useful when your data is not sorted.
- (a) True.
 - (b) False.
13. **Insertions** on the **SquareList** data structure have a **worst-case** runtime complexity of $\mathcal{O}[\sqrt{n}]$. Which of the statements below is correct?
- (a) Retrieving the minimum value has a **worst-case** runtime complexity of $\mathcal{O}[1]$.
 - (b) Retrieval of any element also has a **worst-case** runtime complexity of $\mathcal{O}[\sqrt{n}]$.
 - (c) Deletion of any element has a **worst-case** runtime complexity no better than $\mathcal{O}[n]$.
 - (d) Both (a) and (b).
 - (e) Both (a) and (c).
14. Which of the options below could be used to indicate the end of a linked list?
- (a) The data stored in the last node is an end-of-list flag of some kind.
 - (b) The **next** member of the last node is set to **NULL**.
 - (c) An external pointer pointing to the last node is used.
 - (d) All of the above methods.
 - (e) None of the above methods.

15. Which of the following are limitations of the array-based implementation of the **ADT List**?
- (a) Element retrieval is done in constant time.
 - (b) The list will have a fixed maximum size determined at compile time.
 - (c) Insertions into the list have a **worst-case** runtime complexity of $\mathcal{O}[n^2]$.
 - (d) Both (a) and (b).
 - (e) Both (b) and (c).
16. A **stack** can be implemented using
- (a) an array.
 - (b) a singly-linked list.
 - (c) a doubly-linked list.
 - (d) all three data structures above.
 - (e) none of the data structures above.
17. Suppose I gave you a homework assignment in which you have to simulate automobile traffic coming into a busy intersection. Which data structure would be most appropriate to model the arrangement of cars arriving at the intersection along a given road?
- (a) A linked-list-based stack.
 - (b) A linked-list-based queue.
 - (c) An array-based stack.
 - (d) An array-based queue.
 - (e) Any implementation of a list is fine — there's no need for a stack or a queue.

Part 2. **SHORT-ANSWER** section. You may use **ONLY** the test's last sheet (front and back pages) for **scratch work**, and your final answers to this section must be **SHORT, CLEAN, COHERENT, LEGIBLE**, and written **ONLY** in the spaces provided, or your test will **NOT** be graded. **NO** exceptions will be made.

I will **not** look at your scratch work and **nothing** there will be graded. **Do NOT** remove the scratch pages.

Unless indicated otherwise, each question is worth **1 point** and **NO partial credit** is given. No negative points are given for wrong answers. Please note that you will **NOT** get any credit for leaving questions unanswered.

1. [3 points] Write the C++ implementation of a function `int min(int a[], int length)` which takes as arguments an array of integers, `a`, and its length, `length`, and returns the **smallest** integer stored in the array. Use **ONLY** the boxed space provided below (it may appear on the next page) and please write **LEGIBLY**. Do any scratch work on the scratch pages, **NOT** here!

ANY WORK ON THIS PAGE WILL BE CONSIDERED SCRATCH WORK AND WILL **NOT** BE GRADED. **NO** EXCEPTIONS WILL BE MADE.

Answer Key for Exam A

Part 1. MULTIPLE-CHOICE section. Please choose **only one** of the alternatives provided for each question and remember to mark your answers on the scantron form because **ONLY THOSE** will be considered. If scantron forms are not being used in this test, then make sure to **CIRCLE** your chosen answer.

Each question is worth **1 point** and **NO partial credit** is given. No negative points are given for wrong answers. Please note that you will **NOT** get any credit for leaving questions unanswered.

1. The operation `enqueue()` affects
 - (a) the top of the queue.
 - (b) the front of the queue.
 - (c) the back of the queue.**
 - (d) both the front and the back of the queue.
 - (e) either the front or the back of the queue, but not both at the same time, depending on the user's needs.
2. **FIFO** describes the behavior of a **queue**.
 - (a) True.**
 - (b) False.
3. Consider the following code fragment, where `lptr` is a pointer to an instance of a class implementing the **ADT List**. Assume that the list elements are indexed in the same way an array is.

```
while ((lptr -> getLength()) > 0)
    lptr -> deleteAtIndex(0);
```

The code fragment above:

- (a) deletes the first element of the list.
- (b) leaves the list empty.**
- (c) goes into an infinite loop if the list has more than one element.
- (d) has a **worst-case** runtime complexity of $\mathcal{O}[n]$.
- (e) None of the above.

4. Consider the following code fragment, where `lptr` is a pointer to an instance of a class implementing the **ADT List**. Assume a pointer implementation with no **tail** pointer, but with a **length** member variable. Also assume that the list elements are indexed in the same way an array is.

```
while (( lptr -> getLength() ) > 0)
    lptr -> deleteAtIndex( ( lptr -> getLength() ) - 1 );
```

The code fragment above has a **worst-case** runtime complexity of:

- (a) $\mathcal{O}[\log_2(n)]$.
 - (b) $\mathcal{O}[n]$.
 - (c) $\mathcal{O}[n^2]$.**
 - (d) $\mathcal{O}[n^3]$.
 - (e) None of the above.
5. The **worst-case** runtime complexity of **linear search** when applied to a problem of size n is
- (a) Logarithmic.
 - (b) $\mathcal{O}[n]$.**
 - (c) $\mathcal{O}[n \log_2(n)]$.
 - (d) $\mathcal{O}[n^2]$.
 - (e) Constant.
6. **Black-box** testing of an ADT is a testing procedure in which you don't have access to the ADT's
- (a) source code.**
 - (b) interface.
 - (c) axioms.
 - (d) All of the above.
 - (e) None of the above.

7. What is the correct **increasing** ordering of the following runtime complexities: $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^n]$, $\mathcal{O}[n]$, $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^2]$? Recall that $n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$.
- (a) $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n]$, $\mathcal{O}[n^2]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^n]$.
 - (b) $\mathcal{O}[n^n]$, $\mathcal{O}[n]$, $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^2]$.
 - (c) $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^n]$, $\mathcal{O}[n]$, $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^2]$.
 - (d) $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^2]$, $\mathcal{O}[n!]$, $\mathcal{O}[n^n]$.**
 - (e) $\mathcal{O}[\log_2(n)]$, $\mathcal{O}[n]$, $\mathcal{O}[n \log_2(n)]$, $\mathcal{O}[n^2]$, $\mathcal{O}[n^n]$, $\mathcal{O}[n!]$.
8. Deleting the **last** element of a **singly**-linked list with a **tail** member variable has a **worst-case** runtime complexity of
- (a) $\mathcal{O}[n \log_2(n)]$.
 - (b) $\mathcal{O}[n]$.**
 - (c) $\mathcal{O}[n^2]$.
 - (d) $\mathcal{O}[\log_2(n)]$.
 - (e) $\mathcal{O}[1]$.
9. A **stack** is implemented using a statically defined array of size 100. After the stack is instantiated, 31 pushes and 17 pops are performed on it (not necessarily all pushes before all the pops — assume that they could have been done in any order). Which of the statements below is correct after these operations are performed?
- (a) The stack has 100 elements.
 - (b) The stack has 14 elements.
 - (c) The order in which the operations are performed is important.
 - (d) Both (a) and (c).
 - (e) Both (b) and (c).**
10. An algorithm with **worst-case** runtime complexity proportional to $\log_2(n)$ _____ than another algorithm with **worst-case** runtime complexity proportional to n .
- (a) is always faster
 - (b) is always slower
 - (c) may be faster
 - (d) may be slower**

11. A **List Iterator**

- (a) is an object with responsibilities similar to that of an array index.
 - (b) lets you move from node to node without requiring you to know the details of how the list or the node structures are implemented.
 - (c) is also an ADT.
 - (d) Both (a) and (b).
 - (e) All of the above.**
12. The **binary search** algorithm is useful when your data is not sorted.
- (a) True.
 - (b) False.**
13. **Insertions** on the **SquareList** data structure have a **worst-case** runtime complexity of $\mathcal{O}[\sqrt{n}]$. Which of the statements below is correct?
- (a) Retrieving the minimum value has a **worst-case** runtime complexity of $\mathcal{O}[1]$.
 - (b) Retrieval of any element also has a **worst-case** runtime complexity of $\mathcal{O}[\sqrt{n}]$.
 - (c) Deletion of any element has a **worst-case** runtime complexity no better than $\mathcal{O}[n]$.
 - (d) Both (a) and (b).**
 - (e) Both (a) and (c).
14. Which of the options below could be used to indicate the end of a linked list?
- (a) The data stored in the last node is an end-of-list flag of some kind.
 - (b) The **next** member of the last node is set to **NULL**.
 - (c) An external pointer pointing to the last node is used.
 - (d) All of the above methods.**
 - (e) None of the above methods.
15. Which of the following are limitations of the array-based implementation of the **ADT List**?
- (a) Element retrieval is done in constant time.
 - (b) The list will have a fixed maximum size determined at compile time.**
 - (c) Insertions into the list have a **worst-case** runtime complexity of $\mathcal{O}[n^2]$.
 - (d) Both (a) and (b).
 - (e) Both (b) and (c).

16. A **stack** can be implemented using
- (a) an array.
 - (b) a singly-linked list.
 - (c) a doubly-linked list.
 - (d) all three data structures above.**
 - (e) none of the data structures above.
17. Suppose I gave you a homework assignment in which you have to simulate automobile traffic coming into a busy intersection. Which data structure would be most appropriate to model the arrangement of cars arriving at the intersection along a given road?
- (a) A linked-list-based stack.
 - (b) A linked-list-based queue.**
 - (c) An array-based stack.
 - (d) An array-based queue.
 - (e) Any implementation of a list is fine — there's no need for a stack or a queue.

Part 2. **SHORT-ANSWER** section. You may use **ONLY** the test's last sheet (front and back pages) for **scratch work**, and your final answers to this section must be **SHORT, CLEAN, COHERENT, LEGIBLE**, and written **ONLY** in the spaces provided, or your test will **NOT** be graded. **NO** exceptions will be made.

I will **not** look at your scratch work and **nothing** there will be graded. **Do NOT** remove the scratch pages.

Unless indicated otherwise, each question is worth **1 point** and **NO partial credit** is given. No negative points are given for wrong answers. Please note that you will **NOT** get any credit for leaving questions unanswered.

1. [3 points] Write the C++ implementation of a function `int min(int a[], int length)` which takes as arguments an array of integers, `a`, and its length, `length`, and returns the **smallest** integer stored in the array. Use **ONLY** the boxed space provided below (it may appear on the next page) and please write **LEGIBLY**. Do any scratch work on the scratch pages, **NOT** here!

Answer:

```
int min(int a[ ], int length)
{
    int min_value = a[0];
    for (int i = 1; i < length; i++)
    {
        if (a[i] < min_value)
            { min_value = a[i]; }
    }
    return min_value;
}
```