

600.226: Data Structures Midterm 1

Peter H. Fröhlich
phf@cs.jhu.edu

March 10, 2005

Time: 40 Minutes

Start here: Please fill in the following important information using a **permanent pen** before you do **anything** else! Your exam will **not** be graded if you use a pencil or erasable ink on this page.

Name (print): _____

Login (print): _____

Ethics Pledge: With your signature you **certify** the information above and you also **affirm** the following:
“I agree to complete this exam without unauthorized assistance from any person, materials, or device.”

Signature: _____

Date: _____

Instructions: Please read these instructions carefully before you start. **Switch off** your phones, pagers, and other noisy gadgets! You are **not** allowed to have anything but a pen (pencil, eraser) and this exam on your desk. You are **not** allowed to talk to anyone during the exam. If you have a question, please raise your hand **quietly**. You must **remain seated quietly** until all exams have been collected. Remember that you can **not** claim grading errors if you do not use a **permanent** pen for your answers.

Do not open before you are told to do so!

You got _____ out of 40 points.

1 Binary Warmup

(12 points)

For each of the following statements, determine whether it is either **true** or **false**. (1 point each)

1. Bounded stacks—which can become full—should be a subtype of unbounded stacks.
2. The JAVA profiler is useful for the experimental analysis of algorithms.
3. In a (rooted) tree, no node can simultaneously be the root and a leaf.
4. Algebraic specifications of abstract data types provide test cases for implementations.
5. In any tree the number of internal nodes is less than or equal to (“ \leq ”) the number of external nodes.
6. In JAVA the concepts of `interface` and `abstract class` are interchangeable.
7. Specifications for abstract data types are not concerned with the efficiency of implementations.
8. The most specific class in a generalization (or inheritance) hierarchy is at the root.
9. A (rooted) tree of 14 nodes must be of height greater than 2.
10. In JAVA generic types, i.e. the `T` in `Iterator<T>`, must be reference types.
11. An algorithm that requires $O(n \log n)$ time is always faster than one that requires $O(n^2)$ time.
12. On average Bubble Sort performs fewer assignments than Insertion Sort.

2 Tough Choices

(8 points)

For each of the following questions, circle **one** answer out of the choices given. (2 points each)

1. Consider the level of **detail** available to clients in the form of **implementations** (e.g. source code for classes), **interfaces** (e.g. fully abstract classes), and **specifications** (e.g. algebraic ones). Which of the following seems most appropriate ($A < B$ means “A provides less detail than B”)?
 - (a) Interface $<$ Implementation $<$ Specification
 - (b) Interface $<$ Specification $<$ Implementation
 - (c) Implementation $<$ Interface $<$ Specification
 - (d) Specification $<$ Interface $<$ Implementation
 - (e) None of the above.
2. What is the **best-case** asymptotic complexity of **binary** search on a sorted, doubly-linked list (individual node objects linked by “next” and “previous” pointers)?
 - (a) $O(n^2)$
 - (b) $O(1)$
 - (c) $O(\log n)$
 - (d) $O(n)$
 - (e) $O(n \log n)$
3. Consider the three “simple” sorting algorithms for arrays: Bubble Sort, Selection Sort, and Insertion Sort. You start with the array [3,7,4,1,2]. After **three** iterations of the **outer** loop, you have the array [1,3,4,7,2]. Which algorithm are you running?
 - (a) InsertionSort
 - (b) SelectionSort
 - (c) BubbleSort
 - (d) All of the above produce that array after three iterations.
 - (e) None of the above.
4. Consider the “order” of complexity classes from “fastest” to “slowest” in the intuitive sense. Which of the following seems most appropriate ($A < B$ means that if $f \in A$ holds then $f \in B$ also holds)?
 - (a) $O(n) < O(\log n) < O(n \log n) < O(2^n)$
 - (b) $O(1) < O(n) < O(\log n) < O(n^2)$
 - (c) $O(1) < O(n) < O(n \log n) < O(2^n)$
 - (d) $O(1) < O(\log n) < O(n^2) < O(n \log n)$
 - (e) None of the above.

3 Short Answer

(8 points)

For each of the following questions, answer in **one to three** sentences, the shorter the better. (2 points each)

1. Consider the algebraic specification of ADT Array we developed. Why is $\text{get}(\text{put}(a, i, t), i) = t$ **not** an axiom? What is the correct axiom? **Explain!**
2. Consider **implementing** the ADT Array as a JAVA class. The operations on Array are `new`, `get`, `put`, and `length`. Can **all** of these operations be implemented in such a way as to take **at most** $O(1)$ asymptotic time? If so, how? If not, why? **Explain!**
3. Here's how **level-order** tree traversal works: Put the root into a queue; while the queue is not empty, take a node n from the queue, put n 's children into the queue, and then perform an operation on n . What kind of traversal do you get by replacing the queue with a **stack**? **Explain!**
4. We spent quite some time on designing the interfaces for `Stack<T>`, `BoundedStack<T>`, and `UnboundedStack<T>`. Why did we settle on this peculiar design? **Explain!**

4 Algebraic Specification

(12 points)

Let's specify an abstract data type (ADT) Buffer, which is remotely similar to the ADT Variable. Like Variable, it supports the operations put and get. Unlike Variable, however, it can be empty. A new Buffer is empty. You can put only into an empty Buffer, after which it is not empty anymore. You can get only from a non-empty Buffer. You can reset a non-empty Buffer, after which it is empty again. Complete the following algebraic specification.

adt Buffer

uses

Any, _____

defines

Buffer<T: _____>

operations

new: \rightarrow Buffer<T>

empty: _____ \rightarrow _____

get: _____

put: _____ \times T \rightarrow _____

reset: _____ \rightarrow _____

preconditions

get(b): not(_____)

put(b, t): _____

reset(b): _____

axioms

empty(new())

empty(_____)

not(_____)

_____ (put(b, t)) = _____

This page is intentionally **mostly** blank in case you run out of space elsewhere. If you ended up here early, please go over **everything** again and remain seated **quietly**! Make sure that the title page is filled out correctly and in **permanent** pen. Maybe you want to "rewrite" your **answers** in permanent pen as well?