

600.226: Data Structures

Midterm 1

Peter H. Fröhlich
phf@cs.jhu.edu

October 7, 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): _____

Student ID (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. Algebraic specifications of abstract data types describe syntactic as well as semantic properties.
2. In JAVA the expression "A" + 10 + 25 evaluates to "A1025" and not to "A35".
3. An algorithm that requires $O(\log n)$ time is always faster than one that requires $O(n)$ time.
4. Algebraic specifications of abstract data types provide test cases for implementations.
5. Object-oriented polymorphism means that the sending object decides which method to invoke.
6. Lists should be preferred over arrays if the amount of data to be stored is known in advance.
7. In JAVA the concepts of `interface` and `abstract class` are interchangeable.
8. Experimental analysis compares abstract algorithms instead of concrete implementations.
9. In JAVA generic types, i.e. the `T` in `Iterator<T>`, must be reference types.
10. On average Selection Sort performs fewer assignments than Bubble Sort.
11. In JAVA exceptions should be used to indicate failures of partial abstract data type operations.
12. The most specific class in a generalization (or inheritance) hierarchy is at the root.

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) Implementation $<$ Interface $<$ Specification
 - (c) Interface $<$ Specification $<$ Implementation
 - (d) Specification $<$ Interface $<$ Implementation
 - (e) None of the above.
2. What is the **best-case** asymptotic complexity of **linear** search (in an array or a list)?
 - (a) $O(n)$
 - (b) $O(2n)$
 - (c) $O(1)$
 - (d) $O(\log n)$
 - (e) None of the above.
3. 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) $O(n) < O(1) < O(\log n) < O(n^2)$
 - (b) $O(1) < O(n) < O(\log n) < O(n^2)$
 - (c) $O(1) < O(\log n) < O(n \log n) < O(n^2)$
 - (d) $O(1) < O(\log n) < O(n^2) < O(n \log n)$
 - (e) None of the above.
4. Consider the three “simple” sorting algorithms for arrays: Bubble Sort, Selection Sort, and Insertion Sort. You start with the array [3,4,7,1,2]. After **three** iterations of the **outer** loop, you have the array [1,2,3,7,4]. Which algorithm are you running?
 - (a) BubbleSort
 - (b) InsertionSort
 - (c) SelectionSort
 - (d) All of the above produce that array after three iterations.
 - (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. You need to implement a `Queue` interface but all you are allowed to use is a `Stack` implementation. Describe briefly how you would implement `enqueue`, `dequeue`, and `front` in terms of `Stack` operations and instances. (Consider the unbounded case only.)
2. Consider a non-empty `PositionList<T> list` on which we perform the following operations:
`T t = list.removeFront(); list.insertBack(t)`. How is the new list configuration related to the old one? Can they possibly be identical?
3. You want to add an `Iterator<T>` with the usual operations (i.e. `get`, `put`, `valid`, `next`, `previous`) to an existing `Vector<T>` (aka `ArrayList<T>`) implementation. Describe briefly how you would implement the iterator.
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?

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?