

CS 14: Data Structures and Algorithms

Quiz 1

Peter H. Fröhlich
phf@cs.ucr.edu

Wagner Truppel
wagner@cs.ucr.edu

January 21, 2003

Time: 30 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): _____

Signature: _____

Now have **two** neighbors **confirm** your identity by **signing** below. Provide **official** identification (your UCR card, driver's license, etc.) if requested. Your exam will **not** be graded without these.

Neighbor 1: _____

Neighbor 2: _____

Instructions: 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 and this exam on your desk. You are **not** allowed to talk to anyone during the exam. If you have a question, raise your hand **quietly** and someone will assist you. You have to **remain seated quietly** until we have collected all the exams. 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!

Score: _____ **out of 20 points.**

1 Warmup

(6 points)

Determine whether the following statements are **true** or **false**.

(1 point each)

1. Abstract data types describe **syntactic** as well as **semantic** properties.
2. Abstract data types are **not** helpful in identifying **test cases** for implementations.
3. The **preconditions** of abstract data type specifications can be **enforced** in implementations.
4. Abstract classes in C++ **must** have at least one **pure virtual** member function.
5. Templates in C++ are resolved and checked at **compile-time** and **link-time**.
6. Exceptions in C++ can **only** be used to indicate **failures** of operations.

2 Basic C++

(7 points)

In this section, you are asked to demonstrate basic C++ programming skills for several problems.

1. Implement a C++ function `even` which, given an integer, returns `true` if the integer is *even* (i.e. it can be divided by 2 without remainder) and `false` otherwise. (2 points)

2. Implement a C++ function `odd` which returns `true` exactly when the function `even` from above returns `false` and vice versa. (2 points)

3. Complete the following C++ function `reverse` which takes an array `a` of length `l` and **reverses it in place** in at most $O(n)$ time. **Hint:** Our solution fits on three lines, but we gave you more space just in case... (3 points)

```
void reverse( int a[], int l ) {  
  
_____  
  
_____  
  
_____  
  
_____  
  
_____  
  
}
```

3 Abstract Data Types

(7 points)

adt Array	aka DynamicArray
uses Any, Integer	
defines Array<T: Any>	
operations new: Integer \times T \rightarrow Array<T> length: Array<T> \rightarrow Integer put: Array<T> \times Integer \times T \rightarrow Array<T> get: Array<T> \times Integer \rightarrow T	
preconditions new(l, t): $0 < l$ put(a, i, t): $0 \leq i < \text{length}(a)$ get(a, i): $0 \leq i < \text{length}(a)$	
axioms length(new(l, t)) = l get(new(l, t), i) = t length(put(a, i, t)) = length(a) get(put(a, i, t), j) = if i = j then t else get(a, j)	

The abstract data type **Array** (see above) does not support **changing the length** of an array. Consider a modified ADT that supports a new operation **resize** to change the length of an array in addition to the existing operations **new**, **length**, **get**, and **put**:

1. What is the **signature** of the **resize** operation? (2 points)
2. What are the **preconditions** for **resize**, if any? (2 points)
3. What are the **axioms** for **resize**? (2 points)
4. Do we have to change **anything** else in the **Array** specification? What? (1 point)