Data Structures: Sample Mid-Term Exam

October 5, 2016

It is important to use good test-taking strategy. Spend a couple of minutes looking over the exam, decide which problems you find easy, and do those first. The answers do not have to be in order in your answer booklet.

Be sure to put your name on the exam booklet. You may keep the exam sheet, if you wish.

Problem 1:

What does the following Java code print:

```java
class A {
    public int key;
    public void f() { key = key+1; }
}

class B extends A {
    public void f() { key = key+10; }
}

public class Problem1 {
    public static void main(String[] args) {
        B P = new B();
        P.key = 0;
        A Q = P;
        A R = new A();
        R.key = 8;
        P.f();
        Q.f();
        R.f();
        System.out.println(P.key);
        System.out.println(Q.key);
        System.out.println(R.key);
    }
}
```

Problem 2:

Label each of the following true or false:
A. An abstract class may not contain any concrete methods (i.e. methods with actual code).
B. An interface may not contain any concrete methods.
C. A concrete class may extend more than one abstract class.
D. A concrete class may implement more than one interface.
E. If C is a concrete class extending abstract class A and f is an abstract method of A, then C must contain a concrete definition of f.
F. If C is a concrete class implementing interface I and f is an abstract method of I, then C must contain a concrete definition of f.
Problem 3

A. What is the problem with the following recursive method to compute \( a^n \), for positive integer \( n \)?

```java
public static double exponent(double a, int n) {
    return a*exponent(a,n-1);
}
```

B. Describe how this should be fixed. Your solution should still use recursion. (The code does not have to work if \( n < 0 \)).

Problem 4:

Consider a singly linked list of ints without header composed of nodes with the following definition:

```java
public class MyList {
    public int value;
    public MyList next;
}
```

Write a recursive method \( L \).runningSum() which replaces the value in each node \( N \) with the running sum of the values in the nodes up to and including \( N \). For instance if \( L \) is the list \([2,3,0,1]\) then \( L \).runningSum() changes this to \([2,5(=2+3), 5(=2+3+0), 6(=2+3+0+1)]\).

Problem 5:

Consider the following two implementations of an unordered list,

1. A singly linked list with a pointer to the first element.
2. A doubly linked list, with pointers to the first and last elements

A. Give an example of an operation that can be carried out in constant time in both implementations.
B. Give an example of an operation that can be carried out in constant time in implementation (2) but requires linear time (time proportional to the size of the list) for implementation (1).
C. Give an example of an operation that requires linear time in both implementations.