Data Structures: Sample Mid-Term Exam
October 4, 2017

Problem 1:

What does the following Java code print:

class A {
    public int key;
    public void f() { key = key+1; }
}
class B extends A {
    public void f() { key = key+10; }
}

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

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 4

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.

Problem 5

A programmer is implementing a singly linked list with header. He uses a generic class `MyList<T>` defined as follows:

```java
class MyList<T> {
    public T value;
    public MyList<T> next;
}
```

He wants to write a method `void addAt(T value, int k)` which adds the element `value` into the list at position `k`, using 0-based indexing. For instance, suppose that initially variable `l` is the list of Strings ["Let", "them", "eat", "cake"]. If you now call `l.addAt("chocolate",3);` then `l` will become ["Let", "them", "eat", "chocolate", "cake"]. If you next call `l.addAt("Oh," 0)` then `l` becomes ["Oh," "Let", "them", "eat", "chocolate", "cake"] If you give an invalid index, then nothing happens; for instance `l.addAt("tomorrow", 100)` has no effect, but does not crash.

Write the code for the `addAt` method.