Sample Final

This sample final contains the kinds of questions that will be on the actual final. It is much longer than the final will be. At least one question on the final will be all but identical to one here. The final is comprehensive: some questions may come from first half of the course. See the sample midterm for examples of such questions.

We will not post solutions to the sample final problems because, in our experience, people will be tempted to look at the solution before completely doing the problem. If anyone wishes to check their answers, they may give us a paper copy for review.

1 Definitions and Motivation

- What is a binary tree? If we process a binary tree in "pre order", which node is processed first? Last?
- What property makes a binary tree a binary search tree?
- Insert the following numbers into a binary search tree (in this order, without rebalancing):
  58,37,25,89,78,42,59,38,90,55,88,36
- Give the pre-order, in-order, and post-order traversals of the resulting binary search tree from the previous question.
- Why is it hard to implement an iterator (internal or external) for a binary search tree, even before we consider remove(), and why is it even harder once we consider remove? Explain by suggesting possible implementations.
- How does a hashtable get close to constant-time access?
- Compare and contrast linear probing and chained hashing. Which one has “graceful degradation” and why? (and what is it?)
- What about double hashing? How does it improve over linear probing? What extra requirements are there? Why?
- Why does a hash table with linear probing need a “no object” value or a parallel array of booleans? Give an example of the problem that can happen if we have neither of these techniques? Which technique is more space efficient? Why?
- When writing an iterator for hash tables with chaining, many people find it awkward to implement remove since we didn’t have dummy nodes. Why do you think we don’t use dummy nodes for the buckets?
- When can the remove() method of an iterator be called? What element does it remove? Why have the restriction?
- Whenever a caller or client modifies a library collection class, all iterators are made invalid (with the exception of the iterator through which the change was made, in the case of remove). Give a reason for this rule in the case of an array list (using a dynamically sized array, in which an iterator has a pointer to the array) and in the case of list (using linked list nodes, in which case the iterator points to one or more list nodes).
- When is a linked list a better choice than a dynamic array to implement a container class?
- What is the adjacency matrix technique for implementing graphs? The edge-list technique?
- What is a “sparse” graph? What implementation technique for graphs is inefficient for sparse graphs? Explain.
- What is depth-first search? What is breadth-first search?
• When is it good to use @SuppressWarnings("unchecked") and when not?

• Describe the mergesort algorithm. What algorithmic technique does it use? When is this algorithm to be preferred over insertion sort?

• What is XML? Why is it used? What is the difference between an element and an attribute?

• Why is there a method called addAttr in AbstractShape which simply throws an exception? What design pattern is it a part of? What role does it play?

2 Reading Code

Read the solutions to Homeworks 1-14 and answer the following questions: THESE ARE FOR A PREVIOUS SEMESTER. IMAGINE SIMILAR QUESTIONS!

1. Homework #1
   (a) Why should move() never call paintContents? What could happen wrong if we did so?
   (b) Why should paintContents() never call move() ? What could happen wrong if we did so?
   (c) Why does Location not define a setX method?

2. Homework #2
   (a) What is the connection (if any) between _data.length and _manyItems ?
   (b) Why does DiskSeq.append call ensureCapacity every time it is called? Isn’t that inefficient?
   (c) Why does remove require a for-loop ?
   (d) Why do we throw an exception in parseString has an error? Why not simply print an error message?

3. Homework #3
   (a) Why do we show an error message if parseInt throws an exception? Why don’t we just fix our bug?
   (b) What is the difference between _done and _stopped ? Can we be done when we’re not stopped (still moving) ? Can we be stopped if we’re not done?
   (c) In CarControlSeq, why does the iterator function return this ? (Why not make it a void method?)

4. Homework #4
   (a) Why does _wellFormed count up the nodes rather than just calling size() ?
   (b) The invariant checker checks whether precursor is null or in the list. What would it mean to be non-null and not be in the list? How could that happen?
   (c) appendAll has an empty for-loop in it. What’s the point?
   (d) And why does it call clone ?
   (e) What does it mean for cursor to be null? (See, e.g. hasNext().)

5. Homework #5
   (a) What about the code means that a triangle can be in only one Group at a time?
   (b) What is the purpose of the Key interface? Why does it not include any code?
   (c) How many keys are used in the program? Can you think of any other possible keys?
(d) removeFirst includes a condition in which along one branch, two pointers are assigned and in the other only one pointer is assigned. This is usually the sign of a bug. Why not in this case? (What is the missing assignment?)
(e) In sortForward, we have the condition:
   ```java
   if (p != t._prev)
   ```
   What does this condition refer to? What would be true if the two pointers were equal?
(f) How can sortForward be very efficient even if started at the front? (It has two nested while loops.)

6. Homework #6
(a) cursor is a “ghost field.” What does that mean?
(b) How does hasNext get its value? How does next() set its value?
(c) appendAll special cases the empty addend. Would the rest of the code work properly for this case? Explain!
(d) The dummy node is supposed to make the coding easier. How would append have to be more complex, if we didn’t have a dummy node?
(e) In clone() why is it necessary to change answer.precursor?

7. Homework #7
(a) How do we avoid ever having null pointers in the list? Why is that desirable?
(b) How is the dummy node created to point to itself?
(c) In add, we have the line:
   ```java
   _tail = _tail.next = new Node<T>(x,_tail.next);
   ```
   Why don’t we simply say _tail = new Node(...). Would there be any difference?
(d) Why do we override the clear() method.
(e) What does LinkedList.this._wellFormed() in the iterator code mean?
(f) How does next() ensure you don’t go around in cycles forever?
(g) In MyIterator.remove there is the line:
   ```java
   _myVersion = ++_version;
   ```
   What does this line do? (It does two things!) If this line were omitted, what problem would result? What if it simply said ++version?
(h)

8. Homework #8
(a) Why does ensureCapacity need both i and j when copying elements from one array to another?
(b) Why does offer reject null? Would anything go bad if it accepted null?
(c) Why doesn’t poll shift all the remaining elements over?
(d) Why does clear allocate a new array?
(e) In the following code:
   ```java
   if (++k >= _data.length) k = 0;
   _data[j] = _data[k];
   ```
   We shift k if it falls off the end, but not j. Why not? What is the relation between j and k?

9. Homework #9
(a) What purpose does _checkInRange have? Why something so complicated?
(b) Why doesn’t add permit two listings to have the same price? What would happen if a different listing (different addresses) with the same price was added?

(c) In remove, there is a second while loop:

```java
while (t.left != null) {
    lag = t;
    t = t.left;
}
```

What is this loop doing? Why?

(d) In addAll, the solution gives both efficient and inefficient code. What is inefficient about the (much cleaner looking) “inefficient code”?

(e) Explain the syntax of the following line and describe why it is doing what it does:

```java
int n1 = add(array[mid]) ? 1 : 0;
```

10. Homework #10

(a) In the iterator, what does the stack of _pending nodes represent?

(b) In hasNext there are three separate cases. Indicate what state the system is in (in terms of the client) that these three cases refer to.

(c) It seems that next() has duplicated code (two similar or even identical while loops). Could the last two cases of next() be combined?

(d) What is happening on the line of remove() marked “overwrite”? Explain!

11. Homework #11

(a) Why do we have dummy listings in the table? When do they show up? When do they go away?

(b) The invariant includes the following code:

```java
if (h > i) {
    if (h <= lastNull || lastNull < i ) return _report(...)
} else {
    if (h <= lastNull && lastNull < i) return _report(...)
}
```

Draw pictures to explain what these conditions mean.

(c) The rehash method may decrease the size of the array. When does that happen?

(d) In several methods, we have lines similar to:

```java
++h;
if (h == _contents.length) h = 0;
```

What is going on? Why?

(e) In add, we have the following condition:

```java
if (l2 == dummyListing) {
    if (place == -1) place = h;
}
```

What is that code doing? Explain!

(f) Why does put sometimes increment _numListings, sometimes _numUsed, sometimes neither (returning early), but never _numUsed by itself? What do these three cases represent?

12. Homework #12

(a) Why is _ropen a different size than _copen?
(b) read says it may throw an IOException but in the code, it only throws ParseException. Why is this accepted by the compiler. If we changed IOException in the method header to ParseException, the code would not compile any more. Why not?

(c) How does findPath know if it has reached the goal?

(d) The method findPath ensures that the stack always includes a path. How does it ever consider an alternate route from a node? Why? What would happen if we didn’t do that?

(e) In the drawing code, we have the following condition:

```
if (_marked[_rows-1][0]) {
    g.fillRect(0, (2*_rows-1)*SQUARESIZE, SQUARESIZE, 2*SQUARESIZE);
    g.fillRect((2*_columns-1)*SQUARESIZE, 0, 2*SQUARESIZE, SQUARESIZE);
}
```

What is this code doing? Why is there a condition?

13. Homework #13

(a) Why does Icon inherit from CenteredShape? Why not Polygon or Triangle?

(b) Why does Triangle inherit from Polygon when Rectangle does not?

(c) Why is CenteredShape abstract whereas Polygon is not?

(d) Why does the constructor for Disk not initialize the radius attribute?

(e) What does Point... mean?

14. Homework #14 (TBA)

Read the following code that finds a path by search through a directed graph:

```java
private Set<Node> visited = new HashSet<Node>();

/**
 * Initialize for a new search
 */
protected abstract void init();

/**
 * Add a search state to be considered later
 * @param s search state to add.
 */
protected abstract void add(SearchState s);

/**
 * Are there any more search states to consider?
 * @return whether there are any more search states
 */
protected abstract boolean hasNext();

/**
 * Return next search state to consider
 * @return
 */
protected abstract SearchState next();

public SearchState find(Node from, Node to) {
    visited.clear();
    init();
    add(new SearchState(from));
    */
while (hasNext()) {
    SearchState s = next();
    Node last = s.last();
    if (last == to) {
        return s;
    } else if (!visited.contains(last)) {
        visited.add(last);
        for (Edge edge : last.edges()) {
            add(s.extend(edge));
        }
    }
}

return null;
}

public class SearchState implements Iterable<Edge> {
    private Node initial;
    private List<Edge> path;

    public SearchState(Node i) {
        initial = i;
        path = new ArrayList<Edge>();
    }

    public Node first() {
        return initial;
    }

    public Iterator<Edge> iterator() {
        return path.iterator();
    }

    /**
     * Return the last node on this path.
     * If the path is empty, we're still in the initial node.
     * @return last node on path
     */
    public Node last() {
        ...
    }

    /**
     * Return a new search state which is like this one except
     * with a path extended by the given edge.
     * @param e edge to add to new state's path.
     * @return new search state.
     */
    public SearchState extend(Edge e) {
        ...
    }
}

1. What role does visited play? What would happen if we moved the call to clear the visited set into the “while” loop?
2. How should `add` and `next` be implemented to achieve depth-first search? breadth-first search? Why?

3  Code

- Given a binary tree class called TreeSet that uses instances of a class called Node, write a method that writes the nodes on an output stream for debugging purposes. Your method should show the parent-child relationships in the tree.

```
class TreeSet <T extends Comparable<T>> {
    private static class Node<T> {
        T data;
        Node<T> left, right;
        Node(T d) {data = d; }
    }
    private Node<T> root;

    private void doPrint(Node n, int indent) { ... }

    public void print() { doPrint(root,0); }

    ...
};
```

It is your task to implement `doPrint`, a helper method for `print`. This member function should print the tree where each element is indented by two spaces more than the parent, and the data should be printed in order.

For example, in the following picture, the tree on the left should be printed in the manner shown on the right:

```
      5
     ↙ ↘
    3 8
   ↙ ↙ ↘
  1 6 9
```

- Implement the Dictionary ADT with a single-linked list without a dummy node.

- Suppose we wished to implement a (directed) Graph ADT using edge lists. Nodes are implemented as (small) integers in the range \([0, n]\) where \(n\) is the number of nodes:

```
public class Graph {
    private int numberOfNodes;
    private ArrayList<ArrayList<Integer> > adjacent;

    public Graph(int size){...}

    public int getNumberOfNodes() {...}
    public void addEdge(int from, int to){...}

    public Iterator<Integer> adjIterator(int node){...}
}
```
This representation doesn’t give a way to indicate edge weight. Why not? What would have to be changed in order to do so?

This data structure permits a node to have multiple edges to the same destination node. Explain how this could be changed by changing the data structure.

Implement the ADT.

Use the ADT to write a recursive depth-first search.

- A bank keeps track of its collection of customer records and collection of accounts with two nested classes Customer and Account classes:

```java
class Customer {
    public Customer(String name) {...}
    public void setAccount(Account a) {...}
    public Account getAccount() {...}
}
class Account {
    public Account(Money initial) {...}
    public void setBalance(Money b) {...}
    Money getBalance() {...}
}
```

For a new customer that does not yet have an account, both a customer record and an account get dynamically allocated, with an initial deposit of $100 (i.e., `Money(100)`). The customer record and the account must be added to their respective collections in the most convenient way.

Write the body of a function that creates a new customer and account (with 100 dollars) using the customer name passed as a string:

```java
Collection<Customer> customers;
Collection<Account> accounts;
...
Customer createCustomerAndInitialAccount(string name)
{
    // {Your code here!}
}
```

- For the same Bank as above, write a `totalDeposits` function that computes the total amount of money in all the accounts. Assume that `Money` has a method:

  ```java
  void add(Money m);
  ``

Make sure you don’t modify any of the accounts!

```java
Money totalDeposits()
{
    Money total = new Money(0);
    // Your code here!

    // System.out.println("The total amount is " + total);
    return total;
}
```
• How can an array be used to implement a Stack ADT? Please complete the class declaration, then give the member function implementations separately:

    class Stack<T> {
        T[] contents;
        int used;

        private T[] makeArray(int n) {
            contents = (T[])new Object[n];
        }

        public Stack() { ... }
        public boolean isEmpty() { ... }
        public void push(T x) { ... }
        public T pop() { ... }
    };
int hash(K key) { ... } // return value in range [0,contents.length)
}

You can assume the existence of boolean _report(String s). You don’t need to check that there are no duplicates.

• Write a routine to check the invariant of a hashtable using linear probing. Assume the following data structure:

class HashSet<K> ... {
    K[] contents;
    int numEntries;

    K nullObject = (K) new Object(); // fill-in for null
    K noObject = (K) new Object(); // place holder for removed entry

    int hash(K key) { ... } // return value in range [0,contents.length)
}

NB: make sure you handle the result of probing. You don’t need to check that there are no duplicates. Hint: you don’t need to treat nullObject specially. (Why not?) (NB: This question by itself is too hard for a final exam.)

• Implement insertion sort / merge-sort / quick-sort on an array or on a linked list.

4 Debugging

• Consider a drawing program that uses a class called Canvas to display the following classes:

  A Shape class...
  
  class Shape {
      // draw the shape on a canvas
      public void draw(Canvas c) {} 

      // translate the shape by vector in d
      public void move(Delta d) {
          throw new UnsupportedOperationException("move not implemented");
      }
  }

  A Line class derived from Shape...
  
  class Line extends Shape {
      private Point point1, point2;

      public Line(Point p1, Point p2) {
          point1 = p1;
          point2 = p2;
      }

      // @Override
      public void drawShape(Canvas c) { ... }
  }
@Override
public void move(Delta d) {...}
}

A Square class derived from Shape ...

class Square extends Shape {
  public Square(Point c, int s) {...}

  // @Override
  public void drawShape(Canvas c) {...}

  public void move(int dx, int dy) {...}

  private Point center;
  private int size;
}

The test driver code for Shape, etc. looks like this:

... List<Shape> shapes = new ArrayList<Shape>();

shapes.add(new Square(new Point(10,10),4));
shapes.add(new Line(new Point(0,0), new Point(30,30)));

Canvas canvas = new Canvas(40,40);

for (Shape s : shapes) {
  s.draw(canvas);
}

canvas.dump(System.out);

shapes.get(0).move(new Delta(5,0)); // move Square

When the program is run, the canvas never gets anything drawn on it, and then when we try to move
the square, it aborts. Why? Assume the method bodies with ... are correctly implemented, and
further assume Canvas works correctly. How should the Shape class be modified to avoid errors related
to its' subclasses?

- We have written the following code to implement cloning of a Set implemented with binary search
trees (with parent pointers):

```java
class Set<T> {
  private static class Node<T> {
    Node<T> parent;
    T data;
    Node<T> left,right;
    Node(Node<T> p, T d) {
      parent = p;
      data = d;
    }
  }
```
private Comparator<T> comparator;
private Node<T> root;
private int numItems = 0;
...

@ SuppressWarnings("unchecked")
public Set<T> clone()
{
    Set<T> result;

    try
    {
        result = (Set<T>) super.clone();
    }
    catch (CloneNotSupportedException e)
    {
        // This exception should not occur. But if it does, it would probably
        // indicate a programming error that made super.clone unavailable.
        // The most common error would be forgetting the "Implements Cloneable"
        // clause at the start of this class.
        throw new RuntimeException
               ("This class does not implement Cloneable");
    }

    result.root = new Node<T>(null,root.data);
    result.root.left = root.left;
    result.root.right = root.right;

    return result;
}

This code works fine when the set has exactly one element, but crashes if it is empty, and strange
things sometimes seem to handle if there is more then one element. For instance the invariant might
fail suddenly at the start of a public method.

1. Why does this code crash if the set is empty?
2. Why doesn’t it work correctly when there is more than one node?
3. Give example client (outside) code which will cause an invariant error to happen. Explain why.

• Assume the class SortedList is just a sequence of values that are kept sorted in ascending order.
Consider the following version of SortedList#locate for a linked list implementation:

class SortedList<T> {
    static class Node {
        T data;
        Node pNext;
    };
    Node pHead;
    ...
/** Return (0-based) position of item in sorted list.
 * In other words, return number of items before this one in the list.
 * Return -1 if the item isn’t found.
 * preconditions: none
 */
public int locate(T item) {
    Node p = pHead;
    int count = 0;

    while (p.pNext != null && p.pNext.data != item) {
        p = p.pNext;
        count++;
        if (p.data == item) return count;
    }

    return -1;
}

The code does not work:
1. If we ask for the location of something in an empty list, we get a null pointer exception segmentation fault. Explain why this happens.
2. In all other cases, whether not the item is found, it returns -1. Explain why this happens.
3. Fix both problems by neatly editing the code.

• Someone else working on the same method tried the following:

public int locate(T item) {
    Node p = pHead;
    int count = 0;
    while (p.pNext != null && p.pNext.data != item) {
        p = p.pNext;
        ++count;
        if (p.data == item) return count;
    }

    return (p.pNext == 0) ? -1 : count;
}

If you call locate() on an empty SortedList, the program dies with a null pointer exception. If the item you’re looking for is at the end of the list, the method returns -1. Otherwise, the correct value is returned. What is wrong? Fix the problem(s).

• An XMLEntity class includes the following methods:

    public void addAttribute(String name, String value) { ... } // warn
    public void addXMLEntity(XMLEntity value) { }

Someone implemented Item incorrectly:
private int worth;
public void addAttribute(String key, int value) {
    println("Got here!");
    if (key == "worth") {
        worth = value;
    }
    super.addAttribute(key,"value");
}

But no matter what they put in the XML, it never printed Got here.

1. They wanted to write: super.addAttribute(key,value) but Eclipse said that was an error. Putting quotes around "value" worked. Why is this the wrong solution, in any case?
2. What is the connection between this “fix” and the problem that Got here never gets printed?
3. Even when this is fixed, we get the message
4. Everything compiled, but when we tested it on some test cases, we got the messages:
   Unknown attribute: worth
   and all objects have zero worth when we sell them. What is the problem?
5. After that problem is solved, items now have non-zero worth (if specified in the XML), but we still get the warning messages. Why?
6. Fix the code by neatly editing it, or by rewriting it.

- The following series of classes was designed using inheritance:

```java
class Person {
    private int id;
    private String firstName, lastName;

    public Person() {}
    
    public Person(int id, String firstName, String lastName) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFormattedName() {
        return firstName + " " + lastName;
    }
}

class Student extends Person {
    private double gpa;

    public Student(int id, String firstName, String lastName, double gpa) {
        this.gpa = gpa;
    }

    public double getGPA() {
        return gpa;
    }
}
```
public class Teacher extends Person {
    private String title;
    private boolean isTenured;

    public Teacher(int id, String firstName, String lastName,
                   String title, boolean isTenured) {
        this.title = title;
        this.isTenured = isTenured;
    }

    public boolean getIsTenured() {
        return isTenured;
    }

    @Override
    public String getFormattedName() {
        return title + " " + firstName + " " + lastName; // error #1
    }
}

We use the following driver:

public class Main {

    public static void main(String[] args) {
        Teacher a = new Teacher(100, "Mary", "Jo", "Dr.", true);
        Person b = new Student(200, "John", "Doe", 3.8);

        printPersonInfo(a);
        printPersonInfo(b);
    }

    public static void printPersonInfo(Person p) {
        System.out.println(p.getFormattedName());
        if (p instanceof Student) {
            System.out.println("GPA: " + p.getGPA()); // error #2
        } else if (p instanceof Teacher) {
            System.out.println("Tenured: " + p.getIsTenured()); // error #3
        }
    }
}

1. We get the following compiler errors:

   **error #1** The fields firstName and lastName are not visible.

   **error #2** The method getGPA is undefined for the type Person.

   **error #3** The method getIsTenured is undefined for the type Person.

   Explain each of these error messages.

2. Fix the compiler errors by only changing the lines marked with errors. Either edit the code neatly or place the fixes below:

3. After the compiler errors are fixed, the program runs and prints the following unexpected output:
Dr. null null
Tenured: true
null null
GPA: 3.8

Why? Please explain!

4. Fix the problems.