EE 422C – Midterm Exam – Summer 2016

I promise that all work on this test is my own, that I have received no outside assistance on it, and that I am adhering to the university honor code.

Your Name________________________________ Your UTEID___________
Date________

Problem Number	Question	Points Possible	Points Off	G	RG
1	Short Answers	36			
2	Recursion	24			
3	Scanner	15			
4	Polymorphism	26			
5	Generics	17			
TOTAL POINTS OFF:					
SCORE OUT OF: 120					

Instructions:
1. Please turn off your cell phones.
2. You have 120 minutes to complete the test.
3. You may not use a calculator.
4. There is scratch paper at the end. You may ask for additional scratch paper if required. If your actual answer is on the scratch paper, state that fact clearly and legibly in the space below the question.
5. When code is required, write Java code.
6. You may break problems up into smaller methods. (In other words you may add helper methods wherever you like.)
7. You may not use packages that are not part of the standard Java distribution.
8. Style is not evaluated when grading, but neatness and legibility are required. Use indentation to help the graders read your code.
9. The proctors will not answer most questions. Write down any assumptions that you need to make to resolve your doubts.
10. When you finish, show us your UT ID, turn in the exam and all scratch paper. Use the stapler we have if required.

Problem 1 – Short Answers (36 points)

For all parts of this problem, if the code does not compile, write CE. If there is a runtime error, write RE. If there is an infinite loop, write IL.

Questions A and B use the following method.

```
          public void foo(int[] vals){
               vals[vals.length - 1] += 3;
               vals = new int[2];
               vals[0]++;
          }
```

A. What is output by the following code when method a is called?

```
public void a(){
     int[] dataa = {2, 3};
     foo(dataa);
     System.out.println(Arrays.toString(dataa));
}
```

____

B. What is output by the following code when method b is called?

```
public void b(){
     int[] dataa = new int[0];
     foo(dataa);
     System.out.println(Arrays.toString(dataa));
}
```

____

For parts C-E, evaluate the expression.

C.

```
(32 < 54) || ('a' > 'x') ||
("hello".equals(("Hello").toLowerCase()));
```

____

D.

```
3.14 / 2 + 99 / 100
```

____

E.

```
(char)('a' + 3 + 'H' - 'h')
```

____

F. What is the output printed upon running this program?

```
public class Foo {
    public static void main (String[] args) {
        Sounds cat = new Sounds();
        Sounds dog = new Sounds();
        for (int i = 0; i < 10; i++) {
           cat.cry();
           dog.cry();
        }
        System.out.print(cat.barks + ", " +  dog.barks + ", " +
cat.meows + ", " +  dog.meows);
    }
}

class Sounds {
    static int barks = 0;
    int meows = 0;
    public void cry() {
        barks++;
        meows++;
    }
}
```

Consider these classes for the remaining questions in this section:

```
     public class Computer {
          public int memory() { return 20; }
          public boolean hasBattery() { return false; }
          public String toString() { return "s: " + memory();}
     }

      public class Laptop extends
           Computer { public int memory()
           { return 100;}
           public boolean hasBattery() {return true; }
      }

      public class Desktop extends
           Computer { public int memory()
           { return 10; }
           public String getName() { return "desk"; }
      }

      public class Notebook extends Laptop {
          public boolean rotate () { return true;}
      }
```

`G.` Consider the following statements. For each, answer if it is legal or if it causes a syntax error or runtime error.

`Object objg = new Desktop();` ______________________________

`Laptop cg = new Computer();` ______________________________

`H.` Consider the following statements. For each, answer if it is legal or if it causes a syntax error.

`Computer rh = new Desktop();` ______________________________

`Desktop mh = new Laptop();` ______________________________

For each code snippet below, state the exact output to the screen. **If Compile Error or Runtime Error, indicate CE or RE.**

```
I.
 Computer ri1 = new Computer();
 Computer ri2 = new Computer();
 System.out.print(ri1 == ri2);
```

```
J.
Desktop mj1 = new Desktop();
Desktop mj2 = new Desktop();
System.out.print(mj1.equals(mj2));
```

```
K.
Desktop mk1 = new Desktop();
System.out.print(mk1.hasBattery() + " " + mk1.getName());
```

```
L.
Computer rl = new Desktop();
System.out.print(rl.memory() + " " + rl.getName());
```

```
M.
Computer rm = new Laptop();
System.out.print(rm.memory() + " " + rm);
```

```
N.
Laptop cn = new Laptop();
System.out.print(cn.memory() + " " + cn.hasBattery());
```

O.
```
Computer ro = new Computer();
Laptop co = new Laptop();
Desktop mo = new Desktop();
int xo = ro.memory() + co.memory() + mo.memory();
System.out.print(xo);
```

P.
```
Notebook np = new Notebook();
System.out.print(np.size*np.size);
```

Q.
```
Laptop lq = new Notebook();
System.out.print( ((Notebook)lq).rotate() );
```

R.
```
Computer cr = new Laptop();
System.out.print( ((Notebook)cr).rotate() );
```

P1:

Problem 2 – Recursion (24 points).

(a) Below is the code for an empty `LinkedList` that is null-terminated. Write a method for the `LinkedList` class called

```
           public boolean find (LLNode<T> start, T value)
```

that returns `true` iff the value value is somewhere in the part of the list from the `start` node to the end. You can assume that `start` is part of the list. Use recursion for this method, with no `for` or `while` loops. Note that a static inner class may be used just like a public class described in a different .java file.

```
public class LinkedList<T> {
      private LLNode<T> head;
      private LLNode<T> next;

      static class LLNode<T> {
            T data;
            LLNode<T> next;
            public LLNode (T value) {
                  this.data = value;
                  this.next = null;
            }
      }
      public LinkedList() {
            head = new LLNode<T>(null);
      }
      public void insert(T data) {
            // ... some code  here, you don't write it
      }
      public boolean find(LLNode<T> start, T value) {
            // pre: start is a node in a LinkedList<T>
            // replace the below line with your code.
            return false;
      }
}
public boolean find(LLNode<T> start, T value) {
```

For (b) and (c): The code below shows a BinarySearchTree. Each node of the tree has 0-2 children, with the invariant property that all the nodes attached to the left child of a node N (if a left child exists) have a value smaller than or equal to the N own value, and all nodes attached to the the right child have a larger value than N's own. If a node has no left child, say, it will be a null. A sample tree is shown.

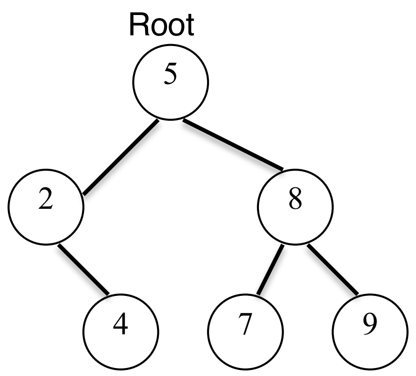
```
public class BST {
      BSTNode overallRoot;

      static class BSTNode {
            Integer data;
            BSTNode left;
            BSTNode right;
            public BSTNode (Integer val) {
                  this.data = val;
            }
      }

      public BST(Integer val) {
            overallRoot = new BSTNode(val);
            overallRoot.left = null;
            overallRoot.right = null;
      }

      // Other methods here to insert a node, remove a node etc.

      public boolean find(BSTNode root, Integer value) {
            //  your code goes here
      }
}
```



Write a find method *using recursion*

```
                  public boolean find (BSTNode n, T value)
```

that returns true iff the value `value` is in the list. For full credit, your method must not search the whole tree by default, but must use the binary search tree's invariant to search only a portion of the tree.

(b) Write the base case or cases for the recursive method, with the correct conditions and return value.

(c) Continue the code by writing the recursive case that runs when the base case conditions fail.

P2:

Problem 3 – Scanner (15 points):
Write a static method `printNonDuplicates(Scanner s)` that takes in a `Scanner` object that is connected to a `File` object (containing printable ASCII text) as a parameter, and prints out all words that are read from the `File`, printing duplicate words only once. The words may be printed in any order, one word on each line. The `Scanner` connected to a `File` behaves exactly as a `Scanner` connected to the keyboard i.e. it scans through white-space separated words in the `File`. When the last word is read by the `Scanner`, subsequent calls to `hasNext()` return false.

For full credit, your program should run in O(N) time, N being the number of words in the file. Slower approaches will yield partial credit. You may use any data structures from Java `Collections`.

For example, if the file is
```
Of shoes and ships and sealing wax,
And cabbages and kings.
```

The method might print the line below.
```
And
sealing
cabbages
kings.
Of
shoes
ships
and
wax,
```

Write your method below:

```
printNonDuplicates(Scanner s) {
```

P3:

Problem 4 – Polymorphism (26 points)
In the following questions, the classes `Rectangle`, `Triangle`, and `Circle` all extend `Shape`, and `Square` extends `Rectangle`.

(a) Write a static method that takes a `Collection` of `Shape` objects (or a `Collection` of `Shape's` subtypes) as argument, and returns an `ArrayList` of all the objects of type `Rectangle` or its subtypes. For full credit, make your code work even if we add other subtypes to `Rectangle`.

(b) I want a `Collection` to hold `HashSets` of objects of type `Square`, `Rectangle`, and `Shape` only, giving a compile error if a user tries to put in anything else. Which declaration(s) would work? Circle all the correct answers.

```
1. Collection<Shape> c;
2. Collection<? extends Shape> c;
3. Collection<? super Square> c;
4. Collection<Square> c;
```

(c) All `Shape` objects have methods `double perimeter()` and `double area()`. They also have a field `Color color`, and a `Color getColor()` method that returns the `Shape` object's `color` field. `Shapes` cannot be instantiated.

Based on the above information, should `Shape` be best implemented as
an abstract class
a concrete class
an interface

Explain your answer in 1-2 sentences.

(d) All `Shape` objects have a `clone()` method that makes a copy of `this` and returns the same type of object. The copy is a new object, and has data that is separate from the original (not just pointers to the original's data fields). Write a `clone` method for the `Shape` class if that is possible based on your answer to (c) (if not, simply write the declaration -- the header without the body -- and declare the method abstract).

(e) Write a `clone` method for the `Circle` class, which has 3 fields –

```
double radius
Point center
```

and the inherited field `Color color`.

All the class fields have clone() methods where new objects are returned, with data fields also being new, and not just pointers to the original data fields.

P4:

Problem 5 – Generics (17 points)

In class we implemented lists that store every value in the list explicitly. However, what if most of the elements of the list equal the same value? Consider the following list:

```
 0   1   2   3   4   5   6   7   8   9    10 11   12 13 14 15 16    //
position [A, B, A, A, A, A, A, A, A, A, AAA, A,
     B, A, A, C, A] // element
```

Storing all those A's seems like a waste of space. In a ***sparse list***, only the elements not equal to the default value are stored explicitly. The default value is set when the list is created and does not change for a given list. Internally if we use a native array as our storage container we must also store the position of each element, because the position in the array does not necessarily equal the position in the list.

Consider the following internal representation of the list shown above. Each element in the array is a `ListElem` object that stores one element of data and the position of that element in the list. `(position, non-default element)`

The elements not equal to the default element, are stored in the array in ascending order based on their position in the list.

```
     0            1         2          3      index
in array [(1, B),  (10, AAA),  (12, B),  (15, C)]
size of list = 17
elements stored = 4
All elements not stored explicitly equal A for this list.
```

Complete the `get(int pos)` method for a `SparseList` class. Consider these examples using the list shown above.

```
list.get(0) -> returns A

list.get(1) -> returns B

list.get(11) -> returns A

list.get(15) -> returns C

list.get(16) -> returns A
```

Here is the `ListElem` class:

```
public class ListElem<E> {
   public ListElem(int position, E data) // create element
   public E getData() // return data of this element
   public int getPosition() // return position of this element
   public void setPos(int pos) // change position of this element
   public void setData(E data) // change data of this element
```

The properties of the `SparseList` class are:

- the internal storage container is a native array of `ListElem` objects
- there may be extra capacity in the native array
- only elements not equal to the default element are stored explicitly
- the non-default elements are stored at the beginning of the array in ascending order based on their position in the list
- the size of the list, the number of elements stored explicitly in the array, and the default value are stored in separate instance variables
- any elements in the ListElem array that are not referring to data-containing elements of the list are set to `null. nulls` may be present throughout `ListElem.`
- the default list value never equals `null.`

```
public class SparseList<E> {

      private ListElem<E>[] values; private int sizeOfList;

      // All values not stored explicitly in values equal
      defaultValue.
      // defaultValue never equals null.
      private E defaultValue;

      // Number of elements stored explicitly in values.
      // The elements are stored at the beginning of the
      array.
      // This value could be 0 even if sizeOfList > 0 indicating that
      // every element in the list is the default value. The
      // sizeOfList - elementsStored = number of nulls in ListElem.
      private int elementsStored;
```

```
/*    pre: 0 <= position < size() of list
      post: Return the element at the given position in this list.
                   This list is not altered as a result of this method.
*/
     public E get(int pos)
```

P5:

Scratch paper