

1.00/1.001

Introduction to Computers and Engineering Problem Solving

Final Exam / December 21, 2005

Name:	
Email Address:	
TA:	
Section:	

You have 180 minutes to complete this exam. For coding questions, you do not need to include comments, and you should assume that all necessary files have already been imported. Use of electronic devices (laptops, cell phones, calculators etc.) is not allowed.

Good luck!

<i>Question</i>	<i>Points</i>
Question 1	/16
Question 2	/ 24
Question 3	/ 36
Question 4	/ 24
Total	/ 100

Question 1: Short answer questions (16 pts):

1. A catch block can throw the exception it is handling a second time.

TRUE FALSE

2. The following code will not compile because the stream is not closed.

```
public void test() {
    try
    {
        FileWriter writer = new FileWriter( f );
    }
    catch ( IOException e )
    {
        System.out.println("caught IOException");
    }
}
```

TRUE FALSE

3. The runtime of insertion sort $O(n^2)$ is always longer than the runtime of quicksort $O(n \log n)$ on the same set of elements

TRUE FALSE

4. Assume the following code fragment is part of a `main()` method that is part of a correctly structured class.

```
int i = 0;
double j=2;
i = 2*j +1;
System.out.println(i);
```

Will the above code compile? _____

If yes, then what is the final value of `i` that will be printed on the screen? _____

5. Assume the following code fragment is part of a `main()` method that is part of a correctly structured class.

```
int[] intArray = new int[3];
for(int i=0;i<=3;i++) {
    intArray[i] = i+1;
}
```

The above code will run without any errors.

TRUE FALSE

6. The following code will compile

```
public class A
{
    public int i;
    public static void main (String args[])
    {
        i = 5;
    }
}
```

TRUE

FALSE

7. What is the output from the following program?

```
public class ConfusingStrings {
    public static String confused(String s)
    {
        return s += "This is confusing.";
    }

    public static void moreConfused (String s)
    {
        s += "This is even more confusing";
    }

    public static void main(String[] args) {
        String s = "Simple String. ";
        System.out.println(s);

        s = confused(s);
        System.out.println(s);
        moreConfused(s);
        System.out.println(s);
    }
}
```

8. The short answer questions below refer to the following class definition:

```
public class Name {
    private String first;
    private String last;

    public Name (String f, String l ) {
        first = f;
        last = l;
    }

    public boolean equals( Object b ) {
        if ( b instanceof Name ) {
            Name nb = (Name) b;
            return ( first.equals( nb.first ) &&
                    last.equals( nb.last ) );
        } else
            return false;
    }
}
```

- a. Name inherits a hashCode() method from Object.

TRUE

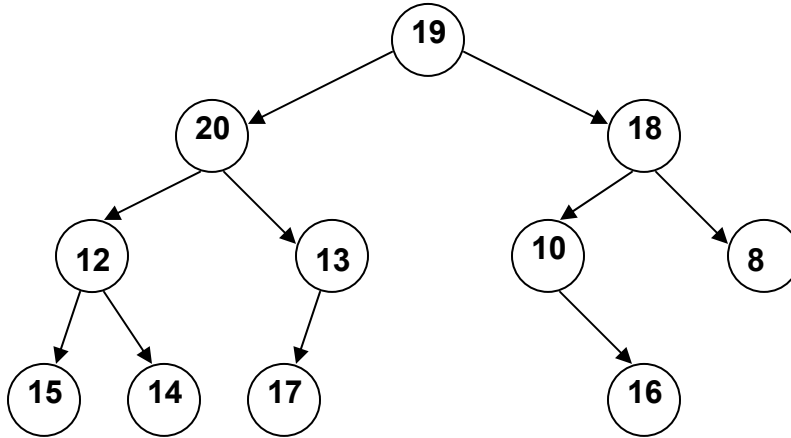
FALSE

- b. What will the following main method print?

```
public static void main( String [] args ) {
    Name n1 = new Name( "Albert", "Einstein" );
    Name n2 = new Name( "Albert", "Einstein" );
    if ( n1.equals(n2))
        System.out.println( "Yes " );
    else
        System.out.println( "No" );
    if ( n1.hashCode() == n2.hashCode())
        System.out.println( "Yes " );
    else
        System.out.println( "No" );
}
```

Question 2 (24 pts)

The next set of questions will be based on the following tree.



Question 2.1

List the value of the nodes of the above tree using post-order traversal.

Question 2.2

Write a recursive algorithm for post-order traversal of the tree. The basic skeleton of a Node class is listed below. Fill in your code in the body of the `traverse` method of the `Traversal` class so that it will call `printNode()` on each node in post-order.

```
public class Node
{
    public Node getLeftChild()
    {
        //implementation hidden
    }

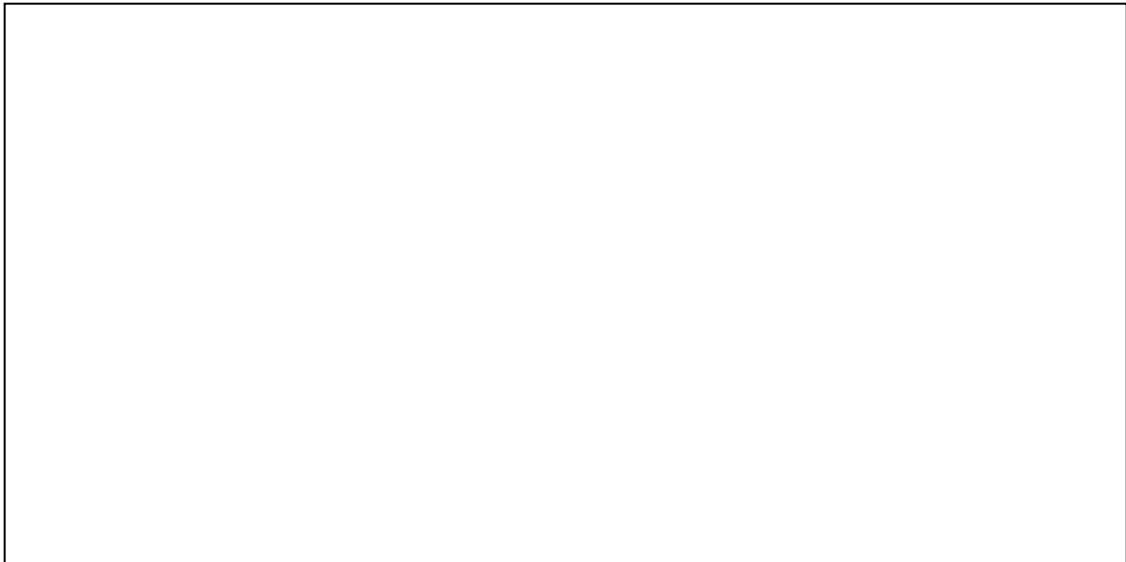
    public Node getRightChild()
    {
        //implementation hidden
    }

    public void printNode()
    {
        // implementation hidden
    }

    //rest of implementation hidden
}
```

```
public class Traversal {

    public void traverse (Node n)
    {
```



```
    }
}
```

Question 2.3

Take your post-order list of nodes from Question 2.1 and insert them (in that order) into a binary search tree using the algorithm described in class. Draw the tree below. List the in-order tree traversal to check your result.




```
public class Movies {  
  
    //this variable will store the path of the text file  
    String filePath;  
  
    public Movies(String filePath){  
        this.filePath = filePath;  
    }  
    /* This method checks if the String "movie" is contained in the  
    * movie file (which remember is specified by the filePath  
    * variable).  
    */  
    public boolean haveMovie(String movie){
```

```
        /*  
        * Question 3.2 Complete this method.  
        * You will need to:  
        * a. open the file stream so you can read it  
        * b. read the file and check if the movie is contained in it.  
        *    return true if the movie is in there and false otherwise  
        * c. make sure to close any streams you open and handle all exceptions properly  
        */
```

```
    }
```

Question 3.3

This method adds the given movie to the file. If the movie is already contained in the file, it throws a `MovieAlreadyThereException`. One strategy to complete the program's goal is to copy the file to a temporary file and rewrite the entire file, including the new movie, back to the original file. You can use the file "c://temp.txt" as your temporary file. Complete the method below. We have provided you with some step by step guidelines, but you are not required to follow them. As long as your method fulfills the stated goals, without any errors, you will receive full credit.

```
public void addMovie(String movie) throws IOException,  
    MovieAlreadyThereException {
```

```
// a. check if the movie you are trying to add is contained in the movie file.  
//     If it is, throw a MovieAlreadyThereException  
// b. open the file stream to read  
// c. open a file stream to write (and create the temp file, "c://temp.txt")  
// d. copy the movie file to the temp file  
// e. close the streams  
// f. open the temp file as an input stream  
// g. open the movie file stream as an output stream  
// h. copy the temp file to the movie file  
// i. close the streams
```

```
}
```

```
public static void main(String[] args) {  
    Movies myMovies = new Movies("C:\\Movies.txt");
```

```
/*
```

```
* Question 3.4 Correctly add the movie named "Big" to the file
```

```
*/
```

```
}
```

```
}
```

Question 4 (24 pts)

You are making an online telephone directory service and storing your data in a linked list. However you notice that your implementation's performance is too slow - searches are taking too long. To try to correct this problem, you decide to build a hash map out of all of the data since you know that hash map lookup performance can be $O(1)$. Because you are using a special data format for your information, you have to write your own hash method for this new data type. Assume that you have already written this hash method. Now you want to test it out.

Question 4.1

While testing you notice that your lookup time to find a telephone directory entry is just as bad as it was with a linked list. What is a likely reason for this? Circle all that apply:

- a) The hash method returns a different value for different keys.
- b) The hash method always returns the same value
- c) The hash method returns drastically different values for slightly different keys.
- d) Accessing the bucket indicated by the hash method takes just as long as a lookup in the linked list, irrespective of the number of entries in the table.

After these unsuccessful tests you decide to abandon your proprietary design and instead use a Java HashMap. You decide to use the String "lastname, firstname" as the key (for e.g. the key for John Smith would be "Smith, John"), and a DirectoryInfo object as the value. Here's the description of the DirectoryInfo class:

```
public class DirectoryInfo {
    public String firstName;
    public String lastName;
    public String telephoneNumber;
    public int zipCode;

    public DirectoryInfo(String fn, String ln, String phone, int
        zipCode) {
        firstName = fn;
        lastName = ln;
        telephoneNumber = phone;
        this.zipCode = zipCode;
    }
}
```


Question 4.4

Now the new system is complete let's analyze it. Mark following statements as TRUE or FALSE.

1) Lookup performance should be about the same whether you have three entries in the hash map or an entry for every person in the United States (assuming you have enough memory to fit that many).

TRUE

FALSE

2) It is possible to have an entry for Bob Jones in Nashua, New Hampshire as well as Bob Jones in Nashville, Tennessee.

TRUE

FALSE

3) If two keys are similar, their entries will be near each other in the `HashMap` because their hash codes will be similar.

TRUE

FALSE

4) If two keys' hash codes are similar, you can be confident that the keys differ by at most a few characters.

TRUE

FALSE

5) If two keys' hash codes are different, you can be confident that the keys are also different.

TRUE

FALSE

HashMap Method Summary

void	clear () Removes all mappings from this map.
Object	clone () Returns a shallow copy of this <code>HashMap</code> instance: the keys and values themselves are not cloned.
boolean	containsKey (Object key) Returns <code>true</code> if this map contains a mapping for the specified key.
boolean	containsValue (Object value) Returns <code>true</code> if this map maps one or more keys to the specified value.
Set < Map.Entry < K , V >>	entrySet () Returns a collection view of the mappings contained in this map.
V	get (Object key) Returns the value to which the specified key is mapped in this identity hash map, or <code>null</code> if the map contains no mapping for this key.
boolean	isEmpty () Returns <code>true</code> if this map contains no key-value mappings.
Set < K >	keySet () Returns a set view of the keys contained in this map.
V	put (K key, V value) Associates the specified value with the specified key in this map.
void	putAll (Map <? extends K , ? extends V > m) Copies all of the mappings from the specified map to this map. These mappings will replace any mappings that this map had for any of the keys currently in the specified map.
V	remove (Object key) Removes the mapping for this key from this map if present.
int	size () Returns the number of key-value mappings in this map.
Collection < V >	values () Returns a collection view of the values contained in this map.