King Fahd University of Petroleum & Minerals
College of Computer Science and Engineering
Information and Computer Science Department
ICS 201 – Introduction to Computing II
Summer Semester 2011-2012 (113)

Solution to Final Exam

31st July 2012

Time: 120 minutes

Name: ______________________________________ **StudentID:** ____________

This exam consists of four questions. All questions must be answered.

Question#	Max Marks	Marks Obtained
1	30	
2	30	
3	25	
4	15	
Total	100	

Q. 1 [2*15 = 30 marks] For each of the following statements, fill in the blank with the correct answer: (Write your answers on the next page in the space provided).

#	Statement
1	If class A is an ancestor of class B, then class B can be called a __*descendent*________ of class A.
2	All variables declared in an interface are **public**, **final** and _*static*________.

3	Complete the following code for a **compareTo** method for a Student class. Assume the Student class contains a double variable **gpa** significant to two places of decimal. The **compareTo** method orders the students according to ascending value of gpa.
	public int compareTo(Object o) {
	Student s = (Student) o;
	double myGpa = this.gpa; double hisGpa = s.gpa;
	return ____*((int) ((myGpa – hisGpa) * 1000))*______; }

4	Sorting Algorithms for arrays of primitive types are based on the ___*quicksort*___ algorithm.
5	In an applet, write code to change the background color to blue below: _*getContentPane().setBackground(Color.blue);*
6	It is possible for a recursive method to have more than one base case. (True/False)? _*True*___
7	In Java, unchecked exceptions are descendents of the class _*RuntimeException*_____.
8	Inner and Outer Classes have access to each other's ____*private*______ members.
9	A program with a _*checked*______ exception will not compile unless the exception is handled using a try-catch block or declared.
10	An Anonymous Inner class has no __*name / constructor*________________.
11	A __*throws*______ clause in a method header is used to indicate that this method throws an exception.
12	An abstract method has no method body, and ends with a _*semicolon*____________.
13	A __*ListIterator*_____________ interface is used to move from the back to the front of an ordered list using hasPrevious() and previous() methods.
14	It is not possible to call the ____*iterator.remove()*_________ method of an iterator twice, without invocating **iterator.next()** in between.
15	Write code to draw an oval of radius 100 **centered** at the point (100, 100). *g.drawOval(0, 0, 200, 200);*

Write your answers here:

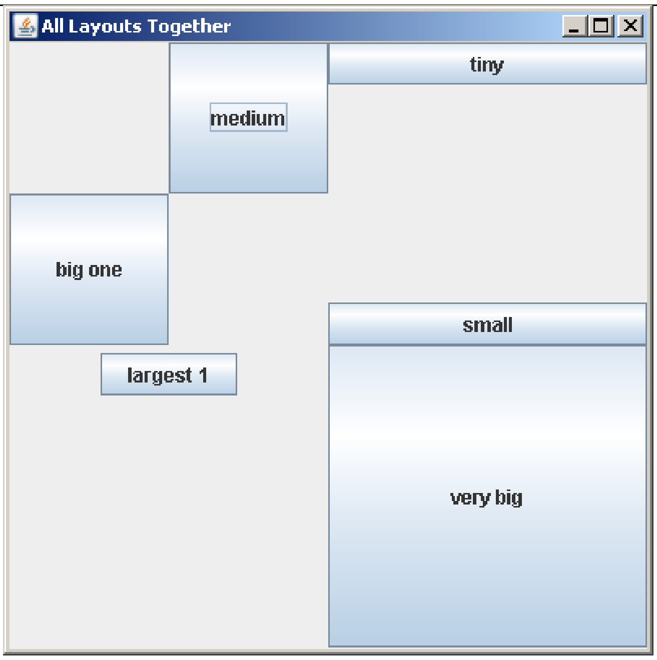
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Q. 2 [**8+7+7+8 = 30 marks**] For each of the following programs, find the output:

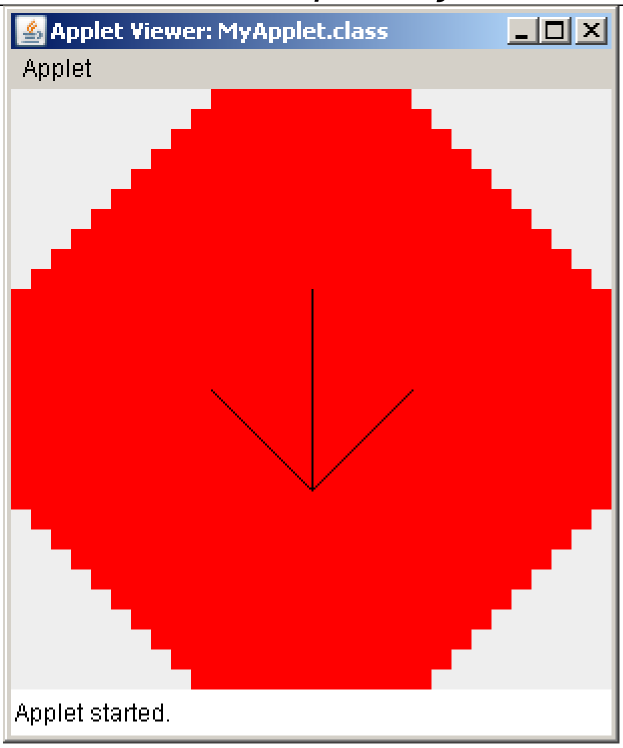
```
public class ExamRec {
      public static void main(String[] args) {
            System.out.println(unknownFunction(-4));
      }

      public static boolean unknownFunction(int x) {
            System.out.print("u"+x+", ");
            if(x == 0)
                  return true;
            else if(x > 0)
                  return brotherFunction(1 - x);
            else
                  return brotherFunction(x - 1);
      }

      public static boolean brotherFunction(int y) {
            System.out.print("b"+y+", ");
            if(y > 0)
                  return (!(unknownFunction(y)));
            else
                  return (!(unknownFunction(-y)));
      }
}
```

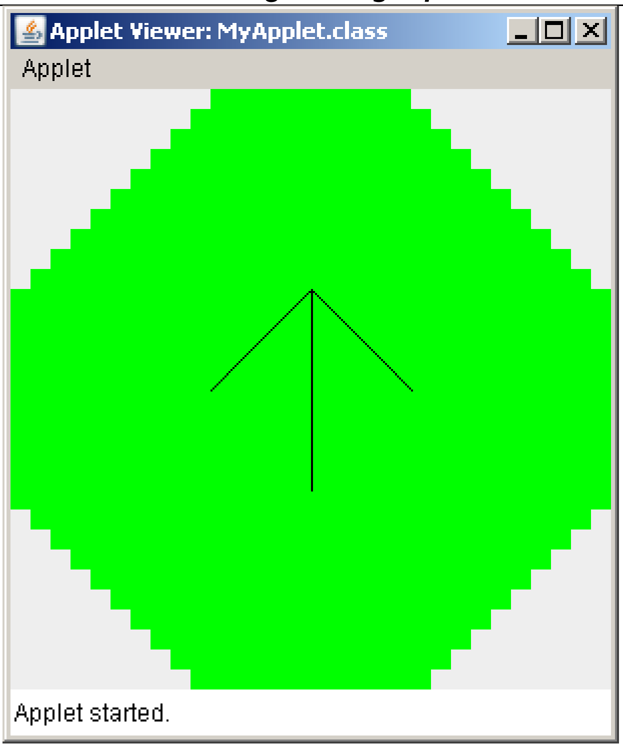
u-4, b-5, u5, b-4, u4, b-3, u3, b-2, u2, b-1, u1, b0, u0, true

```
import java.awt.*; import javax.swing.*;

public class MyProg extends JFrame {
      private JButton b1, b2, b3, b4, b5, b6; private JPanel p1, p2, p3;

     public MyProg() {
            b1 = new JButton("tiny"); b2 = new JButton("small");
            b3 = new JButton("medium"); b4 = new JButton("big one");
            b5 = new JButton("very big"); b6 = new JButton("largest 1");

            p1 = new JPanel(); p2 = new JPanel(); p3 = new JPanel();
            p1.setLayout(new BorderLayout());
            p1.add(b1, BorderLayout.NORTH); p1.add(b2, BorderLayout.SOUTH);

            p2.setLayout(new GridLayout(2, 2));
            p2.add(new JPanel()); p2.add(b3); p2.add(b4); p2.add(b5);

            p3.setLayout(new FlowLayout()); p3.add(b6);

            setSize(400, 400); setLayout(new GridLayout(2, 2));
            add(p2); add(p1); add(p3); setVisible(true);
      }

      public static void main(String[] args) {
            MyProg mp = new MyProg();
      }
}
```



```
import java.awt.*; import java.awt.event.*; import javax.swing.*;

public class MyApplet extends JApplet implements Runnable {
      boolean ok = true; Color color;
       public void init() { Thread t = new Thread(this); t.start(); }
       public void paint(Graphics g) {
            int var; super.paint(g);

            if(ok) { color = new Color(255, 0, 0); var = 200; }
            else { color = new Color(0, 255, 0); var = 100; }

            g.setColor(color);
            for(int i = 100; i > 0; i-=10) g.fillRect(i, 100-i, 300 - 2*i, 10);
            g.fillRect(0, 100, 300, 100);
            for(int i = 0; i < 100; i+=10) g.fillRect(i, 200+i, 300 - 2*i, 10);

            g.setColor(Color.black); g.drawLine(150, 100, 150, 200);
            g.drawLine(100, 150, 150, var); g.drawLine(150, var, 200, 150);   }
     public void run() {
            while(true) {
                  try { Thread.sleep(500); }
                  catch(InterruptedException e) {}
                  ok = !ok; repaint();
}}} //Construct the figure here assuming each square is 10x10pixels.
//Write the description of the threaded behavior in the adjoining space.
```





The applet alternates between these two states once every 0.5 seconds (or twice a second).

```
import java.util.*;

public class Example {
      public static void main(String[] args) {
            ArrayList<String> sl = new ArrayList<String>(5);
            sl.add("exam");  sl.add("assignment"); sl.add("quiz");
            sl.add("lab"); sl.add("java"); System.out.println(sl);

            ArrayList<String> bList = new ArrayList<String>();
            bList.add(sl.get(0));
            for(int i = 1; i < sl.size(); i++) {
                  boolean done = false;
                  for(int j = 0; j < i; j++)
                        if((sl.get(i).compareTo(bList.get(j))) < 0);
                        else {
                              bList.add(j, sl.get(i));
                              done = true;
                              break;
                        }
                  if(!done) bList.add(sl.get(i));
                  System.out.println(bList);
            }
            System.out.println(bList);
      }
}
```

```
[exam, assignment, quiz, lab, java]
[exam, assignment]
[quiz, exam, assignment]
[quiz, lab, exam, assignment]
[quiz, lab, java, exam, assignment]
[quiz, lab, java, exam, assignment]
```

Q. 5 [25 marks] Design and implement a method

public static ArrayList<String> sortByFrequency (ArrayList<String> input)

in Java that takes an arraylist of strings as input. It then sorts the list based on the frequency of occurrence of each string. For example if the input arraylist of strings is `[computer, mouse, desktop, mouse, computer, mouse, desktop, computer, laptop, computer]` then the output is:

```
[laptop, desktop, desktop, mouse, mouse, mouse, computer, computer,
computer, computer]
```

If two strings have the same frequency of occurrence then they should come in alphabetical order. You may add any extra classes or methods as needed.

```
import java.util.*;

class FreqString implements Comparable {
	String s; int freq;

	public FreqString(String s) { this.s = s;  freq = 1; }

	public void update() {  freq++; }

	public boolean equals(Object o) {
		if(o == null) return false;
		else if(o.getClass() != this.getClass())  return false;
		else return (this.compareTo(o) == 0);
	}

	public int compareTo(Object o) {
		if(o == null) throw new ClassCastException();
		else if(o.getClass() != this.getClass()) throw new ClassCastException();
		else {
			FreqString other = (FreqString) o;
			if(this.freq == other.freq) return this.s.compareTo(other.s);
			else return this.freq - other.freq;
		}
	}

	public String toString() {
		return s +":"+freq;
	}
}
```

```
public class MySort {
	public static ArrayList<String> sortByFrequency(ArrayList<String> as) {
		//This makes an arraylist with unique elements and frequencies
		ArrayList<FreqString> asf = new ArrayList<FreqString>();
		for(int i = 0; i < as.size(); i++) {
			FreqString f1 = new FreqString(as.get(i));
			for(int j = i+1; j < as.size(); j++) {
				if(as.get(i).equals(as.get(j))) {
					f1.update();
					as.remove(j);   j--; //This is done not to skip any element
				}
			}
			asf.add(f1);
		}

		//This sorts that list
		Collections.sort(asf);

		//This 'expands' the condensed list
		int times = 0;
		for(int i = 0; i < asf.size(); i+=times+1) {
			times = 0;
			FreqString f2 = asf.get(i);
			for(int j = 0; j < f2.freq - 1; j++) {
				asf.add(i+j, f2);
				times++;
		}}

		//This converts the list of frequency-strings to a list of strings
		ArrayList lastList = new ArrayList<String>();
		for(int i = 0; i < asf.size(); i++)
			lastList.add(asf.get(i).s);

		return lastList;
	}
	//Note that the main method is not needed for the soln
	public static void main(String[] args) {
		ArrayList<String> as = new ArrayList<String>();
		as.add("computer"); as.add("mouse"); as.add("desktop"); as.add("mouse");
		as.add("computer"); as.add("mouse"); as.add("desktop"); as.add("computer"); as.add("laptop");
		as.add("computer");

		System.out.println(sortByFrequency(as));
}}
```

Q. 6 [15 marks] Write a recursive method **`public static int[] removeDuplicates (int[] a, int index)`** that removes duplicates from a given array of integers. For example if **int[] a** = {2, 1, 2, 1, 5, 3, 3, 5, 6, 6}, then **`removeDuplicates(a, 0)`** returns {2, 1, 3 ,5 ,6}

```
import java.util.*;

public class RemoveDuplicates {
	public static void main(String[] args) {
		int[] a = {2, 1, 2, 1, 5, 3, 3, 5, 6, 6};
		int[] b = removeDuplicates(a, 0);
		System.out.println(Arrays.toString(b));
	}

	public static int[] removeDuplicates(int[] a, int index) {
		if(index == a.length - 1) {
			int[] x = new int[1];
			x[0] = a[a.length - 1];
			return x;
		}
		else {
			int[] b;
			int[] r = removeDuplicates(a, index + 1);
			boolean duplicate = false;
			for(int i = 0; i < r.length; i++)
				if(a[index] == r[i]) duplicate = true;

			if(duplicate) {
				b = new int[r.length];
				for(int i = 0; i < r.length; i++)
					b[i] = r[i];
			}
			else {
				b = new int[1 + r.length];
				for(int i = 0; i < r.length; i++)
					b[1+i] = r[i];
				b[0] = a[index];
			}

			return b;
		}
}}
```