Multiple choice questions: [5 points each]

1. In class we discussed a major cryptographic method: RSA. What does RSA stand for?
   a) Reversible Signature Administration
   b) Rice, Spellings, Armitage
   c) Really Sweet Algorithm
   d) Rivest, Shamir, Adleman
   e) Reducto singulo agnosco

2. A stack uses which access protocol?
   a) FIDO
   b) LIFO
   c) MIPO
   d) FIFO
   e) LIPO

3. The statement int[ ] list = {5, 10, 15, 20};
   a) adds 4 int values to array list
   b) initializes list to have 20 int values
   c) declares and initializes list to have 4 int values
   d) declares list but does not initialize it
   e) causes a syntax error because it does not include “new int[4]” prior to the list of values

4. The following nested loop structure will execute the inner most statement (x++) how many times?
   
   for (int j = 0; j < 100; j++)
   for (int k = 100; k > 0; k--)
       x++;
   a) 100
   b) 200
   c) 10,000
   d) 20,000
   e) 1,000,000
5. Given that s is a String, what does the following loop do?
   
   ```java
   for (int j = s.length(); j > 0; j--)
       System.out.print(s.charAt(j-1));
   ```

   a) it prints s out backwards
   b) it prints s out forwards
   c) it prints s out backwards after skipping the last character
   d) it prints s out backwards but does not print the 0th character
   e) it yields a run-time error because there is no character at s.charAt(j-1) for j = 0

6. Given that an integer value 5 was stored in an Object by the statement
   ```java
   Object obj = new Integer(5);
   ```
   what is the correct way to recover this value?
   a) int val = obj.parseInt();
   b) int val = (int) obj;
   c) int val = (Integer) obj;
   d) int val = ((Integer) obj).intValue();

7. [15 points]  Determine if two stacks are equal recursively, as follows:
   • if s1 and s2 are both empty, then they are equal
   • if either s1 or s2 is empty and the other is not, then they are not equal
   • if neither are empty, then they are equal if the top of s1 equals the top of s2, and the rest of s1 is equal to the rest of s2.

   ```java
   public static boolean eqRecursive(Stack s1, Stack s2) {
       // assuming neither s1, s2, nor contents are null
       if (s1.empty() && s2.empty())
           return true;

       if (s1.empty() || s2.empty())
           return false;

       return (s1.pop().equals(s2.pop()) &&
               eqRecursive(s1, s2));
   }
   ```
public interface RectI {
    /* Determine whether a point (x,y) is within a rectangle. 
     * (The border is considered within the rectangle) 
     */
    public boolean contains( int x, int y);

    /* Compute area of rectangle (length * width) */
    public int area();

    /* Determine whether another rectangle lies entirely within 
     * this one. Hint: Use the contains() method on corners. 
     */
    public boolean containedIn(RectI r);
}

public class Rect implements RectI {
    int x1, x2, y1, y2;
    public Rect ( int xlow, int ylow, int xhigh, int yhigh ) {
        x1 = xlow;
        y1 = ylow;
        x2 = xhigh;
        y2 = yhigh;
    }

    // Test harness
    public static void main( String[] args ) {
        RectI r1 = new Rect(0, 0, 3, 3);
        RectI r2 = new Rect(1, 1, 2, 2);
        RectI r3 = new Rect(1, 1, 4, 4);

        System.out.println("(1,1) contained ... expect true, true: ", r1.contains(1,1), ", " + r2.contains(1,1));
        System.out.println("0,0 contained ... expect true, false: ", r1.contains(0,0), ", " + r2.contains(0,0));
    }
}
System.out.println("r2 in r1 ... expect true: "+ r2.containedIn(r1));
System.out.println("r3 contained in r2 ... expect false: "+ r3.containedIn(r2));

//write the other methods for Rect below

public boolean contains( int x, int y)
{
    return ((x1 <=x) && (x <= x2) && (y1 <= y) && (y <= y2));
}

public int area( )
{
    return (x2-x1)*(y2-y1);
}

public boolean containedIn(RectI r)
{
    // the comments suggest this, even though it
    // cannot compile (x1,y1,x2,y2 not in interface)
    return (contains(r.x1,r.y1) && contains(r.x2,r.y2));

    // the test harness implies
    return (r.contains(x1,y1) && r.contains(x2,y2));

    // we accepted both answers
}

}
9. [15 points] Suppose we have an array of ArrayLists. A constructor is
provided. You will need to provide code for the addToShortest method.
This will take an int, convert it to Integer (so it can then be converted to
Object), and add it to the end of the shortest ArrayList in the array. Hint:
since arrayOfList[i] is itself an ArrayList, you can say things like
arrayOfList[i].size().

```java
public class ManyArrayList {

    public ArrayList[] arrayOfList;

    public ManyArrayList(int size){
        arrayOfList = new ArrayList[size];
        for (int i=0; i<arrayOfList.length; i++)
            arrayOfList[i] = new ArrayList();
    } // constructor

    public void addToShortest(int newVal){

        if (arrayOfList.length == 0)
            return;

        int shrt = 0;
        for (int i=1; i<arrayOfList.length; i++)
            if (arrayOfList[i].size()<arrayOfList[shrt].size())
                shrt = i;
        arrayOfList[shrt].add(new Integer(newVal));

    } //addToShortest

} //ManyArrayList
```
From ArrayList:
public int size()
Returns the number of elements in this list.

public boolean add( Object o)
Appends the specified element to the end of this list.

From Integer:
public Integer(int value)
Constructs a newly allocated Integer object that represents the specified int value.

10. [10 points] Given an array int[] values, write code which will compute (as a double) the average and print it.

   public static void printAverage(int[] values) {
       int sum = 0;
       double result;

       for (int i=0; i<values.length; i++)
           sum += values[i];

       if (values.length == 0)
           result = 0;
       else
           result = ((double) sum)/values.length

       System.out.println(result);
   }
11. [15 points] Write a method which doubles each occurrence of any vowel. If
s="A Hello!", then your method should return "AA Heelloo!". If s = "bee", then
return "beee". You will be provided with a method isVowel, which determines
whether a given character is a vowel.

```java
public static boolean isVowel(char c)
{
    // determines if c is one of 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', or 'U'
    // code omitted to save space
}

public static String doubleVowel(String s) {
    String result = ""; // replace this with StringBuffer if you want
    for (int i=0; i<s.length(); i++) {
        result += s.charAt(i);
        if (isVowel(s.charAt(i)))
            result += s.charAt(i);
    }
    return result;
}
```
public class LinkedListExample
{
    private class Node{

        int info;
        Node next;

        private Node(int info, Node next) {
            this.info = info;
            this.next = next;
        }
    }

    public Node head, tail;

    public LinkedListExample()
    {head = null;
    tail = null;}

    public void addToFront(int val)
    {
        head = new Node(val, head);
        if (tail==null)
            tail = head;
    }

    public void addToRear(int val)
    {
        if (tail==null)
            head = tail = new Node(val, null);
        else {
            tail.next = new Node(val, null);
            tail = tail.next;
        } //else
    }
}
```java
public void removeFromFront()
{
    if (head == tail)
        tail = null;
    if (head != null)
        head = head.next;
}

public void removeFromRear()
{
    if (head == tail) {
        head = tail = null;  // may already be null
        return;
    }

    Node current = head;
    while (current.next != tail)
        current = current.next;

    current.next = null;
    tail = current;
}

public int size()
{
    int count = 0;
    Node current = head;
    while (current != null)
    { current = current.next; count ++ }
    return count;
}
```