CMPUT 115 Section B1
Term test 1

January 5, 2001

Instructions:

- This is a closed book, no notes exam.
- Try to put all of your answers in the space provided.
- The backs of pages can be used for rough work.
- Be sure to write your student id number on each internal page.
- Please do not open the exam until you are instructed to do so.
- Good luck.

First Name:

Last Name:

student id: ____________________

1. [2 Marks] What output will be produced by the following program? Please put your answer to the right of the code.

```
class H {
  protected static int x = 4;
  protected int y = 5;

  public void setY( int newy ) {
    y = newy;
    x = newy*10;
  }
  public String toString() {
    return "x = " + x + ", y = " + y;
  }
  public static void main(String args[]) {
    H h1 = new H();
    H h2 = new H();
    h1.setY( 7 );
    h2.setY( 4 );

    System.out.println( h1.toString() );
    System.out.println( h2.toString() );
  }
}
```

2. [2 Marks] Why is the concept of *information hiding* useful in programming? Recall that we defined information hiding as: *the practice of hiding the implementation details of a data structure.*

3. [3 Marks] Implement the `removeElementAt` method of the Vector class. There is space on the next page to put your answer.

4. [5 Marks] Implement a method for the Vector class called `getRange`. This method should take two ints, and it should return a new Vector that contains the elements from the original vector in the range specified by these two ints (that is from its first argument to one less than its second argument). For example if a vector **v** contains the objects:

   ```
   [ a, b, c, d, e, f ]
   ```

   then the call `v.getRange(2,5)` should return a new vector that contains the objects:

   ```
   [ c, d, e ]
   ```

   There is space on the next page to put your answer.

student id: ____________________

```
public class Vector {
  protected Object elementData[]; // the data
  protected int elementCount;     // # of elements in vector

  public void removeElementAt(int where) {
  // pre: 0 <= where && where < size()
  // post: indicated element is removed, size decreases by 1
```

```
  public Vector getRange( int start, int end ) {
  // pre:  start and end are valid indexes into this vector
  // post: returns a new Vector that contains the elements in this
  //       vector from start to end-1.
```

student id: ____________________

5. [1 Mark] Name one way that an Array is better than a Vector.

6. [3 Marks] What is the **worst-case** time complexity of the Vector class's `insertElementAt` method? Express your answer as a function of **n**, where n is the number of elements in the vector. Be as exact as you can be.

7. [3 Marks] Circle true or false for each of the following:

a) $n^2 + n = O(n)$	T	F
b) $n = O(\log_2 n)$	T	F
c) $\log_2 n = O(n)$	T	F

8. [2 Marks] Consider the following method.

```
public static int g( int n )
{
    if ( n == 1 ) return 4;
    else if (n%3 == 0) return g(n/3) + g(n-1);
    else return g(n-1);
}
```

What is returned by the call `g(7)` to the above method? Recall that `%` is the remainder operator (for example `9%4` is `1`).

student id: ____________________

9. [2 Marks] What is quick sort's worst-case time complexity? Under what circumstances will this happen? Express your answer as a function of **n**, where n is the number of elements in the array being sorted.

10. [2 Marks] Consider the following sorting algorithm.

```
public void insertionSort( Comparable[] data )
// post: objects in data in ascending order
{
   int numSorted = 1;
   int i;
   while (numSorted < data.length)
   {
         Comparable temp = data[numSorted];
         for ( i = numSorted; i > 0; i-- )
         {
               if ( temp.compareTo(data[i-1]) < 0 )
                     data[i] = data[i-1];
               else
                     break;
         }
         data[i] = temp;
         numSorted++;
   }
}
```

How many calls to `compareTo` are made when `selectionSort` is called on an array constructed as follows (there is room for your answers following the code):

```
Comparable[] data = new Comparable[6];
data[0] = new Integer(12);
data[1] = new Integer(6);
data[2] = new Integer(1);
data[3] = new Integer(-5);
data[4] = new Integer(34);
data[5] = new Integer(-5);
```

Number of calls to `compareTo`:________