Points off	1	2	3	4		Total off	Net Score

CS 307 – Midterm 1 – Spring 2009

Your Name_____

Your UTEID _____

Circle yours TA's name: Todd Guhan Xiuming(aka David)

Instructions:

1. Please turn off or silence your cell phones.
2. There are **4** questions on this test.
3. You will 2 hours to complete the test.
4. You may not use a calculator or any other electronic devices while taking this test.
5. When code is required, write Java code.
6. When writing a method, assume the preconditions of the method are met.
7. When writing a method you may add helper methods if you wish.
8. When you complete the test show the proctor your UTID and give them the test and any scratch paper. Please leave the room quietly.

1. (2 points each, 30 points total) Short answer questions. Place your answers on the attached answer sheet. For code sample state the output. If the code would cause a syntax error answer "syntax error". If it would cause an exception answer "exception". If it would result in an infinite loop answer "infinite loop".

A. What is the output of the following client code?

```
public int methodA(int x, int y){
     x++;
     y--;
     return x * y;
}

// client code
int c = 4;
int d = 5;
System.out.println( methodA(c, d) + " " + d );
```

B. What is output by the following code?

```
int[] listOne = {3, 2};
int[] listTwo = {2, 2};
listTwo[0]++;
System.out.println( listOne == listTwo);
```

C. What is the output of the following client code?

```
public void methodC(int[] list){
     list[0]++;
     list = new int[3];
}

// client code
int[] data = {2, 1};
methodC(data);
System.out.println( Arrays.toString(data) ); // prints out elements
```

D. What is the output of the following client code? It uses `method_c` from part C.

```
// client code
int[] data2 = new int[0];
methodC(data2);
System.out.println( Arrays.toString(data2) ); // prints out elements
```

E. Consider the following method header:

```
// pre: str != null
public boolean allVowels(String str){
```

What were the two ways presented in class for checking the preconditions of methods such as `str != null`?

F. What is output by the following code?

```
int x = 4;
int y = 9;
int[][] table = new int[x * x][y / x];
System.out.println( table[3].length );
```

For questions G – O consider the following classes and interfaces.

```
public interface Movable{
      public int getMoveDistance();
}

public class GroundUnit{

      private String name;
      private int value;

      public GroundUnit(){
            this("gen", 10);
      }

      public GroundUnit(String n, int v){
            name = n;
            value = v;
      }

      public void upgrade(){
            name = name + "I";
            value += 10;
      }

      public String toString(){
            return name + " " + value;
      }

      public int getNum(){
            return value;
      }
}

public class Tank extends GroundUnit implements Movable{

      private int moveValue;

      public Tank(String name, int value, int move){
            super(name, value);
            moveValue = move;
      }

      public int getMoveDistance(){
            return moveValue;
      }

      public void upgrade(){
            super.upgrade();
            moveValue += 2;
      }

      public int getNum(){
            return moveValue;
      }
}
```

G. State if the following declarations are valid or invalid (meaning they cause a syntax error). 1 point each

```
Movable m1 = new GroundUnit(); // 1
Movable m2 = new Movable(); // 2
```

H. State if the following declarations are valid or invalid (meaning they cause a syntax error). 1 point each

```
Movable m3 = new Tank("Max", 10, 5); // 1
Tank t1 = new GroundUnit("Van", 10); // 2
```

I. State if the following declarations are valid or invalid (meaning they cause a syntax error). 1 point each

```
Object obj = new Tank("Rex", 5, 5); // 1
Movable m4 = new GroundUnit("Net", 50); // 2
```

J. What is the output of the following code?

```
GroundUnit g1 = new GroundUnit();
g1.upgrade();
System.out.println( g1 );
```

K. What is the output of the following code?

```
Tank t2 = new Tank("Max", 5, 10);
t2.upgrade();
System.out.println( t2 );
```

L. Briefly explain why the following code does not compile.

```
Tank t3 = new Tank();
System.out.println( t3  + " " + t3.getMoveDistance() );
```

M. What is the output of the following code?

```
GroundUnit g2 = new Tank("Snoopy", 10, 4);
System.out.println( g2.getNum() );
```

N. What is the output of the following code when method `n2` is called?

```
public void n1(GroundUnit g){
     g.upgrade();
     g.upgrade();
}

public void n2(){
     GroundUnit gVar = new GroundUnit("Tex", 5);
     n1(gVar);
     System.out.println(gVar);
}
```

O. Briefly explain why the following code does not compile.

```
Movable m5 = new Tank("Star", 2, 4);
m5.upgrade();
```

2. Implementing classes. (20 Points) Create a class to model integers of an arbitrary magnitude. The Java `int` data type is limited to the range -2^{31} to $2^{31} - 1$, approximately -2 billion to positive 2 billion. The Java `long` data type is limited to -2^{63} to $2^{63} - 1$, approximately -9.2×-10^{18} to 9.2×-10^{18}. In this question you will implement part of a class to model arbitrarily large integers similar to the Java `BigInteger` class.

- Declare a `LargeInt` class.
- The class will use a native array of `int`s to store the digits of an arbitrarily large decimal integer.
- The class will store all digits as positive values between 0 and 9 inclusive.
- Much like the `IntList` class we developed in lecture the `LargeInt` class may have extra capacity. When constructing a `LargeInt` include 5 extra elements in the array of `int`s that stores the digits. Include a private class constant to store this value.
- Because the internal array may have extra capacity it is necessary to track the number of digits of the `LargeInt`.
- The class must have a way of storing whether the `LargeInt` is positive or negative.
- The integer 0 is a special case. For this question we will assume 0 is a positive number. When creating the array to store zero there will be 6 elements all equal to 0.
- Create the class so that the least significant digit (the ones place) is stored in element 0 of the internal array. Thus the number -1735 would be stored as follows:

5	3	7	1	0	0	0	0	0

```
isPositive: false
numDigits: 4
```

- The constructor for the `LargeInt` class will have one parameter, a `String`, with a length greater than or equal to 1. All characters in the `String` will be between `'0'` and `'9'` with the exception of the first one which may be a negative sign `'-'` if the `String` represents a negative number. There will not be any leading zeros with the exception of the `String "0"` which represents `0`. You do not have to check the preconditions on the constructor.

  Recall to convert from a `char` that represents a digit to the corresponding `int` value use the expression

  ```
  c - '0'
  ```

  where `c` is a `char` between `'0'` and `'9'` inclusive.

On the next page complete the required portions of the `LargeInt` class including the class header, the private instance variables, the class constant and the constructor with one parameter of type `String`. You decide how to track the sign of the `LargeInt`. You can use whatever `String` methods you want, but no other Java classes.

Complete the required portions of the `LargeInt` class including the class header, the private instance variables, the class constant and the constructor with one parameter of type `String`. You decide how to track the sign of the `LargeInt`. You can use whatever `String` methods you want, but no other Java classes.

3. The `IntList` class. (25 points) In lecture we developed an `IntList` class to represent a list of `int`s to demonstrate encapsulation and the syntax for building a class in Java. As discussed in class the internal storage container may have extra capacity.

Complete a method for the `IntList` class named `getExpansion`. This is an instance method in the `IntList` class that returns a copy of the calling `IntList` object except that each element in the original `IntList` is now repeated a specified number of time.

Here is the method header:

```
/*    pre: numReps > 0
      post: Create a copy of the calling object and expand each element in
      the original list so that it appears numReps times in a row.
      The calling object is not altered as a result of this method call.
*/
public IntList getExpansion(int numReps){
```

Examples of calls to `getExpansion`.

Original `IntList`	Returned `IntList`
`[0, 3].getExpansion(3)`	`[0, 0, 0, 3, 3, 3]`
`[5, 2, 2, 1].getExpansion(2)`	`[5, 5, 2, 2, 2, 2, 1, 1]`
`[0, 3, -2, 4, 5].getExpansion(1)`	`[0, 3, -2, 4, 5]`
`[].getExpansion(5)`	`[]`
`[4].getExpansion(6)`	`[4, 4, 4, 4, 4, 4]`
`[0, 2, 0].getExpansion(4)`	`[0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0]`

You may not use any other methods in the `IntList` class except for the default constructor and the `size` method unless you define and implement them yourself in this question. Recall that this method is in the `IntList` class and so you have access to all `IntList` objects' private instance variables. You may not use objects or methods from other Java classes.

```
public class IntList{

      private static final int DEFAULT_CAP = 10;

      private int[] container;
      private int listSize;

      // creates a new IntList with listSize equal to 0 and
      // a capacity equal to DEFAULT_CAP.
      public IntList()

      // return the size of this list
      public int size()

/*    pre: numReps > 0
      post: Create a copy of the calling object and expand each element in
      the original list so that it appears numReps times in a row.
      The calling object is not altered as a result of this method call.
*/
public IntList getExpansion(int numReps){
// complete this method on the next page.
```

```
/*    pre: numReps > 0
      post: Create a copy of the calling object and expand each element in
      the original list so that it appears numReps times in a row.
      The calling object is not altered as a result of this method call.
*/
public IntList getExpansion(int numReps){
      assert numReps > 0;
```

4. 2D arrays. (25 points) Complete an instance method for the `MathMatrix` class that returns `true` if the there is at least one row in the `MathMatrix` object passed as an explicit parameter whose elements are a **integer** multiple of the values in the given row of the calling (`this`) `MathMatrix` object. Otherwise the method returns `false`.

Consider the following example. Assume these are the cells of the calling `MathMatrix` object.

2	1	4	3	10
3	2	1	2	5
-2	**3**	**5**	**-2**	**2**
12	5	5	1	-6

`cells` of the calling (`this`) `MathMatrix` object.

Assume the specified row in the calling object is 2 whose elements are in bold above. Further assume the following are the cells in the other `MathMatrix` object sent as a parameter to the method.

5	4	2	1	-5
-5	12	5	12	13
6	2	4	1	6
6	**-9**	**-15**	**6**	**-6**
5	6	3	-12	-13
-8	*12*	*20*	*-8*	*8*

`cells` of the other `MathMatrix` object which is passed as an explicit parameter.

This row is a multiple of -3 of row 2 in the above matrix.

This row is a multiple of 4 of row 2 in the above matrix.

Assume row number 2 in the calling object is the specified row. Notice that the values in row 3 in the other `MathMatrix` object are achieved by multiplying row 2 in the calling `MathMatrix` object by `-3`. So in this case the method would return `true`. If one element of row 3 in the other `MathMatrix` had not been equal to the corresponding element from row 2 in the calling `MathMatrix` times `-3` then the that row would not be a multiple of the given row. (For example if the -9 had been -8 instead.) The method still would return `true` though because row 5 in the other matrix is a multiple of 4 of the given row in the first matrix.

Complete the method `hasMultipleOfRow` based on the following assumptions and constraints.

- Recall this is an instance method in the `MathMatrix` class so you have access to all `MathMatrix` object's private instance variables.
- Do not use any other Java classes when completing this method.
- You may not use any other methods in the `MathMatrix` class other than the `numRows` and `numCols` methods unless you define and implement them yourself in this question.
- Recall the number of rows in the `MathMatrix` object equals `cells.length` and the number of columns equals `cells[0].length`. There is no extra capacity.
- `MathMatrix` objects will have at least one row and at least one column per row.
- The `MathMatrix` objects will have the same number of columns.
- `cells` is always a rectangular 2d array.
- None of the elements of either Matrix will equal 0.
- You will not get full credit if your code performs more checks than are necessary.
- Neither the calling object or the explicit parameter are changed as a result of the method.
- You do not need to check the preconditions or create code to deal with cases when the preconditions are not met.

```
public class MathMatrix{

     private int[][] cells;

     public int numRows()

     public int numCols()

     // complete the following instance method.

     /*   pre: other != null, other.numCols() == numCols(),
          0 <= row < numRows(). None of the elements of the calling
          object or other are equal to 0.
          post: Returns true if any row in other is an integer
          multiple of tgtRow in the the calling (this) MathMatrix
          object. Otherwise the method returns false.
     */
     public boolean hasMultipleOfRow(MathMatrix other, int tgtRow){




// more room on next page if needed
```

Scratch Paper

Question 1 answer Sheet

Name_____

A. _____

B. _____

C. _____

D. _____

1.

2.
E. _____

F. _____

1.

2.
G. _____

1.

2.
H. _____

1.

2.
I. _____

J. _____

K. _____

L. _____

M. _____

N. _____

O. _____