Problem 1 [15 points] *General Concepts*

Answer the following questions. Questions with blanks require only one word. Be concise and clear.

1a [1 point] The correct spelling of your COM S/ENGRD 211 instructor's last name is **SCHWARTZ**.

1b [2 points] What is a data structure?

collection that holds information

1c [2 points] What is the difference between a list and a set?

set: collection of unique items, no repeats
list: imposes order on the collection and allows repeated

1d [1 point] Why is an array not a dynamic data structure?

has fixed size after created

1e [1 point] Give one example of a dynamic data structure: **list**.

1f [1 point] Because of **dynamic** binding, Java uses the actual type of an object to access a method.

1g [1 point] Because of **static** binding, Java uses the reference type to access a field.

1h [2 points] Write the output for the following code for when the user runs the program with this command line:

```
> java args args "args" \"args\"
```

```
public class args {
   public static void main(String[] args) {
      for ( int arg = 0 ; arg < args.length ; arg++ )
         System.out.print( args[arg] +" ");
      }
}
```

args args "args"

1i [2 points] Distinguish between a shallow and deep clone. No answers about sheep or cults…just data structures.

shallow: copy of "top" or first reference to data structure
deep: all items in a data structure are copied

1j [2 points] Distinguish between *height* and *depth* in a tree data structure.

height: path from node to a leaf in a tree
depth: path from the root to a node in a tree

Problem 2 [5 points] *Induction*

Use induction to prove that $1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$ is *true*. For full credit, you must clearly state the base case, induction hypothesis, inductive step, and conclusion in your proof.

Base Case:

try n=1
LHS: 1^2 = 1
RHS: 1*(1+1)*(2*1+1)/6 = 2*3/6 = 1
LHS = RHS, so BC is OK

Inductive Hypothesis:

Assume that sum(i^2,i=1..n) = n*(n+1)*(2*n+1)/6 is true

Inductive Step:

See if relationship is true for the k+1 value.

RHS =
(k+1)*(k+1+1)*(2*(k+1)+1)/6
= (k+1)*(k+2)*(2*k+3)/6
= (k+1)*(2*k^2+7*k+6)/6
= (2*k^3+9*k^2+13*k+6)/6

LHS =
sum(i^2,i=1..k+1)
= sum(i^2,1..k) + (1+k)^2
= k*(k+1)*(2*k+1)/6 + (1+k)^2 (substitute hypthosis for sum(i^2,i=1..k))
= (2*k^3+3*k^2+k)/6 + (6*k^2+12*k+6)/6
= (2*k^3+9*k^2+13*k+6)/6

Conclusion:

LHS = RHS, so the hypothesis is indeed true. Hallelujah!

Problem 3 [10 points] *Recursion*

Using *recursion*, complete method **`mod(int n,int d)`**, which returns the remainder, or ***modulus***, of $n \div d$. Note that we do *not* imply integer division of *n* by *d*! You may *not* use the **`Math`** class, division (**`/`**), multiplication (**`*`**), the modulus operator (**`%`**), or any helper methods and classes. Assume that $n \geq 0$ and $d \geq 1$.

```
public class Problem3 {

   public static void main(String[] args) {

      System.out.println(mod(11,4)); // output: 3
      System.out.println(mod(10,5)); // output: 0
      System.out.println(mod(1,7));  // output: 1

   }

   // Return remainder of n divided by d. See problem specifications above:
      public static int mod(int n, int d) {
```

```
      if (n < d)
        return n;
      else return
        mod(n-d,d);
```

```
      } // Method mod

} // Class Problem3
```

Problem 4 [15 points] *Inheritance*, *Subtyping, Fun With Java!*

4a [12 points] In the box on the next page, write the output that the following program will generate. Hint: There are 7 output values.

4b [3 points] On the next page, answer this question: Would removing the **`//`** in front of the statement **`r4.print(1,2)`** cause at an error when recompiling the program? Why or why not?

```
public class Problem4 {
   public static void main(String[] args) {
   A r1 = new B(1);
   A r2 = new C();
   B r3 = (B) r2;
   System.out.println(r3.y);
   I1 r4 = new C();
   // r4.print(1,2);
   }
}

abstract class A {
   public int x = 3;
   public int y = x;
   public A() { print(x+2); }
   public void print() { System.out.println(x); }
   public void print(int x) { print(); }
}

class B extends A {
   public int y = x;
   public B(int x) { x = this.x; print(x+3); }
   public B() { }
   public void print() { super.print(); }
   public void print(int x) { System.out.println(x+1); }
   private void print(int x, int y) { print(x*y); }
}

class C extends B implements I1 {
   C() { super(3); }
   public void print(int x, int y) { print(x-y); }
   public void m1() { }
}

interface I1 { }
```

Problem 4a output:

6
7
6
7
3
6
7

Problem 4b answer:

causes error
method needs to be specfied in interface because the method *name* is bound at compile time

Problem 5 [30 points] *Singly-Linked Lists, Building Lists, Traversing Lists*

Background: A ***circular linked list*** is a linked list for which the tail points to the head. To create such a list, you can create a regular singly linked list, starting with the head. By making the head the next element of the tail, you have formed a circular list. By maintaining a reference to the original head, you have an ***entry node*** that links to the entire circular list.

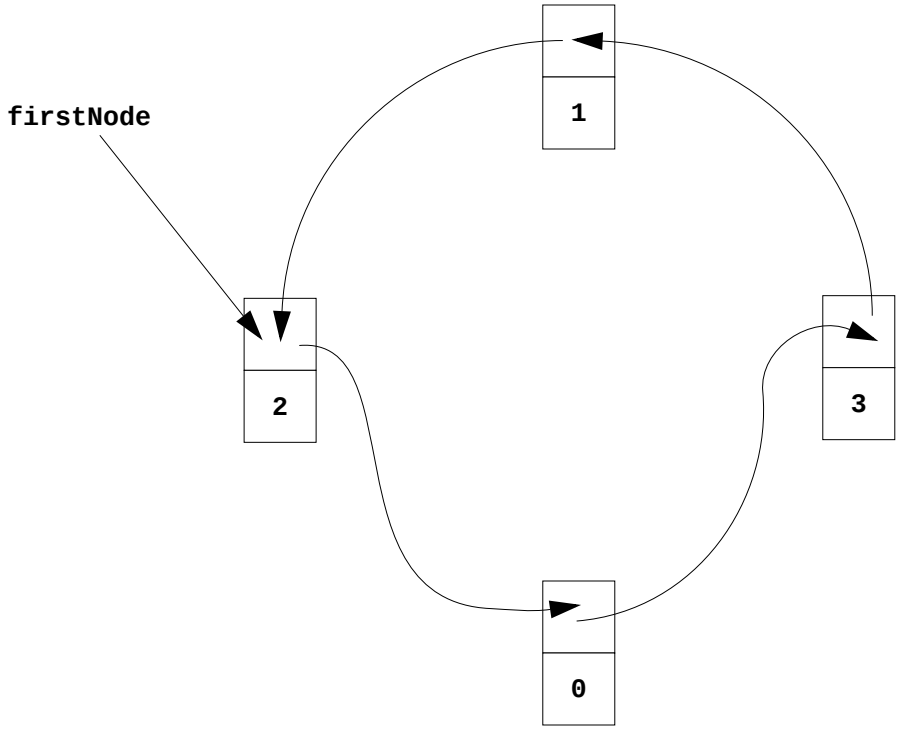
Problem: You need to complete class **`Problem5`** by completing two methods:

- **`createCircle(int nodes)`**. Given a user-input number of **`nodes`**, this method creates a circular linked list composed of **`nodes`** number of nodes with an entry node called **`firstNode`**, which is a class variable in **`Problem5`**.
- **`searchCircle()`**. This method searches for a node with the stored value of **`0`** in the circular linked list that is referenced by **`firstNode`**. If a node with **`0`** is found, the method returns **`true`**. Otherwise, it returns **`false`**. The method assumes the list contains at least one element.

Class **`Problem5`** uses class **`Node`**, which uses class **`MyMath`**. These classes are written at the end of the problem. No credit will be given for using arrays or any API data structure (e.g., **`Vector`**s and **`ArrayList`**s).

Example:

If the user runs the program for 4 nodes as **`> java Problem5 4`**, the program might generate the values that are shown in the circular list below. Since at least one node contains **`0`**, the program would return **`true`**.



The problem continues on the next page.

```
public class Problem5 {

   public static Node firstNode;                // entry node into circular list

   public static void main(String[] args) {
      createCircle(Integer.parseInt(args[0]));  // create list of input # of nodes
      System.out.println(searchCircle());       // report search of circle for 0
   }

   // Create a circular linked list with user-input number of nodes.
   // List uses firstNode as the first node to start the list:
   public static void createCircle(int nodes) {
```

```
firstNode = new Node();
int count = 1;
Node nextNode = firstNode;
  while (count < nodes) {
     nextNode.next = new Node();
     nextNode = nextNode.next;
     count++;
}
nextNode.next=firstNode;
```

```
   } // Method createCircle
```

The problem continues on the next page.

```
   // Search the circular linked list for the integer 0.
   // Return true if found and false if not found.
   // Assume the list is non-empty with at least one node:
   public static boolean searchCircle() {
```

```
Node finger = firstNode.next;
int value = firstNode.value;
while ((value != 0) && (finger != firstNode)) {
  value = finger.value;
  finger = finger.next;
}
return (value==0);
```

```
   } // Method searchCircle

} // Class Problem5

class Node {
   public final int value=MyMath.randInt(0,4);
   public Node next;
} // Class Node

class MyMath {
   // Return a random int between low and high, inclusive:
   public static int randInt(int low, int high) {
      return (int) (Math.random()*(high-low+1)) + (int) low;
   }
} // Class MyMath
```

Problem 6 [25 points] *Trees*, *Expression Parsing*

Background: A ***pretty simple expression*** (PSE) has the following grammar:

- $E \rightarrow int$
- $E \rightarrow (E + E)$

Problem: You will complete a program that parses a user-input PSE into a tree and then prints the contents of that tree, where each operator (+) is written *between* its operands. To do so, finish these methods:

- **`makeTree`**. This method stores the **`root`** of the expression tree and controls the parsing of the PSE.
- **`toString`**. You need to complete this method in each class that implements **`INode`**. Depending on the class, **`toString`** returns a description of the node in the form of a **`String`**. By printing the tree in **`main`**, your program will *print the entire tree*, which gives the desired output, as we have demonstrated in the example session.

Additional Specifications:

- Assume that the user enters only *one* completely legal PSE at the command-window. Do not check for errors.
- Use **`CS211In`**. We have provided a reminder of constants and methods you may need, below.
- All nodes in the tree must implement the **`INode`** interface, which we have also provided.

Example Session:

```
> java PSE
Enter a PSE: ( ( 1 + 2 ) + ( 3 + ( 4 + 5 ) ) )
Result: 1+2+3+4+5
```

Reminder of `CS211In`:

```
interface CS211InInterface {
   int INTEGER = -1, WORD = -2, OPERATOR = -3, EOF = -4;
   int peekAtKind();        // returns "type" (int!) of token w/o "eating" it
   int getInt();            // reads an int and returns it ; else complains
   String getWord();        // reads a word and returns it ; else complains
   char getOp();            // reads an op  and returns it ; else complains
   boolean check(char c);   // is the next thing c?
   void pushBack();         // back up by one token
}
```

```
public class Problem6 {
   private static CS211InInterface fin = new CS211In();
   public static void main(String[] args) {
      System.out.print("Enter a PSE: ");
      System.out.println("Result: "+(new ParseTree(fin)));
      fin.close();
   }
} // Class Problem6

interface INode {
   public String toString();
}
```

The problem continues on the next page.

```
class ParseTree implements INode {
   private INode root;
   public ParseTree(CS211InInterface fin) {
      root = makeTree(fin);
   }
   // Build a PSE:
   private INode makeTree(CS211InInterface fin) {
```

```
 switch( fin.peekAtKind() ) {
  // E -> int
  case CS211In.INTEGER:
    int i = fin.getInt();
    return new IntNode(i);

  // E -> (E1 + E2)
  case CS211In.OPERATOR:
    fin.check('(');
    INode leftExpr = makeTree(fin);
    fin.check('+');
    INode rightExpr = makeTree(fin);
    fin.check(')');
    return new AddNode(leftExpr,rightExpr);

 } // end switch
 return null;
```

```
   } // Method makeTree
```

```
   // Return String of entire tree using root.
   // If tree is null, return empty string:
   public String toString() {
```

```
if(root==null) return "";
return root.toString();
```

```
   } // Method toString()
} // Class ParseTree

class AddNode implements INode {
   private INode leftExpr;
   private INode rightExpr;
   public AddNode(INode leftExpr, INode rightExpr) {
      this.leftExpr=leftExpr;
      this.rightExpr=rightExpr;
   }
   // Return String description of AddNode:
   public String toString() {
```

```
return leftExpr + "+" + rightExpr;
// Why no toString.leftExpr and toString.rightExpr?
// I'm forcing toString to happen by adding a String
// with "+".
```

```
   } // Method toString
} // Class AddNode

class IntNode implements INode {
   private int value;
   public IntNode(int value) {
      this.value = value;
   }
   // Return String description of value:
   public String toString() {
```

```
return ""+value;
```

```
   } // Method toString
} // Class IntNode
```

Bonus: Do not work on these problems until you have thoroughly finished all core-point (required) problems!

B0) [1 bonus point] The answer to this other question (today in lecture 3/6): **22**

B1) [1 bonus point] What is the answer to this question? **17.2**

B2) [1 bonus point] What does *UoS* stand for? You must correctly spell it for full credit.
University of Saskatchewan (pg 242 in DS&SD)

B3) [9 bonus points] Correctly spell the last name of each TA for this course [1 point/correct last name].

Conlon, Fink, Flynn, Kulkarni, Lim,
Lin, Niculescu-Mizil, Qiu, Rosofsky

B4) [1 bonus point] What is the *complete* title of the book that DIS refers to as *DS&SD*?

Data Structures and Software Development in an Object-Oriented Domain: Java Edition

B5) [1 bonus point] Who is Vintersorg?

lead vox in Borknagar; also an eponomously named band and other side projects

B6) [1 bonus point] Which consultant(s) has (have) the most number of hours according to the website?

Kumar

B7) [3 bonus points] What is the correct output for the following program? Yes, it really does work.

```
class X {
   int X;
   X(int X) { this(X,X++); this.X=X; }
   X(int X, int XX) { this.X+=XX+=this.X+X++; }
   int X(int X) { X(this); return X; }
   X X(X X) { X.X+=++x.XX; return new X(X.X); }
} // class X

public class x {
   static int XX;
   public static void main(String[] XXX) {
      X X = new X( (new X(++x.XX)).X );
      X(X.X(X.X));
      X(X.X(X).X);
      X(X(X.X(X)));
   } // method main

   static int X(X X) {return X.X;}
   public static void X(int X) {System.out.println(X);}
} // class x

// Output1: 3

// Output2: 9

// Output3: 13
```