Do not turn this page until you have received the signal to start. (Please fill out the identification section above, write your name on the back of the test, and read the instructions below.)

Good Luck!

This midterm consists of 3 questions on 11 pages (including this one). When you receive the signal to start, please make sure that your copy is complete.

• Legibly write your name and student number on this page. Legibly write your name on the back page of this exam.

• If you use any space for rough work, indicate clearly what you want marked.

• In all programming questions you may assume all input is valid.

• You do not need to write Javadoc.

• You may use a pencil; however, work written in pencil will not be considered for remarking.

# 1: _____/ 8
# 2: _____/13
# 3: _____/16

TOTAL: _____/37
**Question 1.**  [8 marks]

Paola and Rohit are on the CSSU executive and are running a hackathon. They share a subversion repository located at http://markus.cdf.toronto.edu/extra-curricular/hackathon. The master repository is currently empty, and each of them has checked out a local copy on their own laptop.

**Part (a)**  [2 marks]

Rohit has created files `registered.txt` and `scores.txt` in his local copy. What command(s) must he execute to get these files into the master repository?

```
svn add registered.txt
svn add scores.txt
svn commit -m "new files with registered teams and team scores"
```

**Part (b)**  [2 marks]

Paola updates her local copy; then she decides to rename the file `registered.txt` and commit the change to the master repository, as follows:

```
paola$ svn update
Updating '.':
 A   registered.txt
 A   scores.txt
Updated to revision 3.
paola$ mv registered.txt registered-teams.txt
paola$ svn commit -m "renamed file"
```

She thinks everything went well, but when she checks the status of her local copy, she is surprised to see this:

```
paola$ svn status
?   registered-teams.txt
!   registered.txt
```

What went wrong?

She did her "mv" command outside of svn, so subversion didn’t know it had happened. As a result, it wondered what happened to registered.txt, and didn’t know anything about the new file registered-teams.txt.
Correct her sequence of commands above so that they do what Paola intended.

Instead of:
   mv registered.txt registered-teams.txt
Paoloa should have said:
   svn mv registered.txt registered-teams.txt

Part (c) [2 marks]
Suppose after some work on the project, Rohit and Paola each have the latest revision of the master repository in their local copies and there are no local changes. Here is the current content of scores.txt.

Paola: scores.txt
| Team D 510       |
| Mighty Mice 600 |
| Snafu 107       |

Rohit: scores.txt
| Team D 510       |
| Mighty Mice 600 |
| Snafu 107       |

Now, they both edit their local copies of scores.txt:

Paola: scores.txt
| Team D 510       |
| Mighty Mice 600 |
| Weasley Twins   |
| Snafu 107       |

Rohit: scores.txt
| Team D 510       |
| Mighty Mice 600 |
| Snafu 1109       |

Suppppose Paola commits her changes as follows:

paola$ svn commit -m "added a new team"

What is the contents of the master repository now?

It is exactly the same as Paola’s, since Rohit hasn’t committed yet. I.e.,:
| Team D 510       |
| Mighty Mice 600 |
| Weasley Twins   |
| Snafu 107       |

Part (d) [2 marks]
Suppose Rohit now commits his changes as follows

rohit$ svn commit -m "updated Snafu’s score"
Describe what happens.

Rohit’s commit creates a conflict in svn, since his change is incompatible with the master repository.
Question 2. [13 marks]

The following code compiles without errors:

```java
public class Tree {
    public Integer age;
    public String colour = "green";
    public Tree() {
        this.age = 0;
        System.out.println("trunk");
    }
}

public class Dog {
    public Dog() {
        super();
    }
    public void shed() {
        System.out.println("dog hair everywhere");
    }
}

public class DeciduousTree extends Tree {
    String colour = "it changes";
    public DeciduousTree() {
        System.out.println("leaf");
    }
    public void shed() {
        System.out.println("leaves are falling");
    }
}

public class ConiferousTree extends Tree {
    public ConiferousTree () {
        System.out.println("cones");
    }
}
```

Let’s consider some segments of code that might go in a `main` method somewhere in the same package as our code above.

(a) [2 marks] Will the following code compile and run without error? □ yes □ no

```java
Tree t1 = new Tree();
Tree.age = 150;
```

If yes, show the output. If no, explain:

No. The first line is fine, but the second is not. The variable 'age' is an instance variable, and must be accessed through an instance of Tree, not through the class name. For example, we could say 't1.age' 'Tree.age' is only correct if 'age' is static.
(b) [2 MARKS] Will the following code compile and run without error? □ yes □ no

```
Tree t2 = new DeciduousTree();
System.out.println(t2.colour);
System.out.println(((Tree) t2).colour);
```

If yes, show the output. If no, explain:

Yes, and there is lots of output!
  trunk
  leaf
  green
  green

(c) [2 MARKS] Will the following code compile and run without error? □ yes □ no

```
ConiferousTree t3 = new DeciduousTree();
System.out.println(t3.colour);
System.out.println(((Tree) t3).colour);
```

If yes, show the output. If no, explain:

No, because variable t3 must refer to some kind of ConiferousTree object, i.e., a ConiferousTree or some descendant of class ConiferousTree.

(d) [2 MARKS] Will the following code compile and run without error? □ yes □ no

```
Object o = new Dog();
o.shed();
```

If yes, show the output. If no, explain:

No. The second line generates a compile-time error. The compiler cannot confirm that o, which it only knows is an Object, will have a shed method.
(e) [1 MARK] Is any method overridden? If so, name one that is. If not, say "None".

None

(f) [4 MARKS] Somewhere in the same package as our code, we want to write a method called `cleanUp`. It must have one parameter called `thing`. We must be able to pass to `thing` any `Object` that has a method called `shed` that takes no arguments (for example, a `Dog` or a `DeciduousTree`). The signature of our new method will look like this:

```java
public void cleanUp ( )
```

1. We need to define something new in order to make this possible. Write the code to define it:

We can define an interface to describe objects that have a `shed` method that takes no arguments:

```java
public interface Shedder {
    public void shed();
}
```

2. In addition, one or more changes must be made to the classes we defined. Make these changes, directly on the code.

We must declare that `Dog` implements `Shedder` and `DeciduousTree` implements `Shedder`.

3. Finally, fill in the parameter portion of the method signature above to achieve our goal.

```java
public void cleanUp ( Shedder s )
```
Question 3. [16 marks]

A Tuple has a capacity and holds elements in it. Elements are accessed by their index, which starts at 0, and all elements must all be of the same type. Until an element is given a value by method put, it is null. Once given a value, an element can never be changed. Any null values in a Tuple are represented by X when it is printed. This main method demonstrates how to use a Tuple:

```java
public static void main(String[] args) {
    Tuple<String> words = new Tuple<String>(3); // A Tuple with capacity 3
    boolean ok;
    System.out.println(words); // Prints "( X X X )"
    ok = words.put(2, "coming");
    ok = words.put(0, "winter");
    System.out.println(words); // Prints "( winter X coming )"
    ok = words.put(1, "is");
    System.out.println(words); // Prints "( winter is coming )"
    ok = words.put(2, "cold"); // Can't change this element; it already has a value
    System.out.println(ok); // Prints "false"
    System.out.println(words); // Prints "( winter is coming )"
    ok = words.put(49, "far away"); // Can't put something here; it's out of range
    System.out.println(ok); // Prints "false"
    System.out.println(words); // Prints "( winter is coming )"
    Tuple<Integer> numbers = new Tuple<Integer>(15);
    System.out.println(Tuple.longest()); // Prints "15", length of the longest Tuple
    System.out.println(numbers); // Prints "( X X X X X X X X X X X X X X X X X X X )"
}
```

Implement class Tuple, using an array to store its contents. You may define additional instance variables as required. Ensure that the main method above compiles, runs, and produces the output shown. You should assume that the main method is in class Tuple.

Reminder: You must use an array. You may not use an ArrayList, any kind of List, or any class in the Java Collections Framework.
Solution:

```java
public class Tuple<T> {
    private T[] contents;
    private static int longest = 0;

    public Tuple(int size) {
        this.contents = (T[]) new Object[size];
        longest = Math.max(longest, size);
    }

    public boolean put(int i, T item) {
        if (i < 0 || i >= this.contents.length || this.contents[i] != null) {
            return false;
        } else {
            this.contents[i] = item;
            return true;
        }
    }

    public static int longest() {
        return longest;
    }

    @Override
    public String toString() {
        String answer = "( ";
        for (int i = 0; i < this.contents.length; i++) {
            if (this.contents[i] == null) {
                answer += "X ";
            } else {
                answer += this.contents[i] + " ";
            }
        }
        return answer + ")";
    }
}
```
Short Java built-in class descriptions

class ArrayList<E> is not used on this test!

class Math:
   // Contains methods for performing basic numeric operations.
   static int max(int a, int b)
      Returns the greater of two int values.

I don’t think you need anything else from the Java API.

SVN commands
svn add PATH
svn checkout URL
svn commit [PATH] -m MESSAGE
svn delete PATH
svn list [PATH]
svn mv OLD NEW
svn status [PATH]
     Meaning of output:
     A: item is scheduled for addition.
     D: item is scheduled for deletion.
     M: item has been modified.
     ?: item is not under version control.
     !: local copy is missing.
svn update [PATH]
svn resolved

Simplified description of Unix commands
mkdir NAME
   make a new directory called NAME
ls
   list contents of the current directory
mv OLD NEW
   rename file OLD to NEW
cd SUBDIRECTORY
   change directories to SUBDIRECTORY
cat FILE
   display contents of FILE

Total Marks = 37