COP 3530
Data Structures

Midsemester Exam

Name: _____
Email: _____

October 15, 2002

This exam has 3 questions. Each question starts on a new page. Please answer each question on its page. For questions 2 and 3, write your answer on the **facing page.** You should not need more than one page. You may assume `java.util` has been imported. There will be no deductions for lack of commenting. There will be no deductions for lack of import directives. There will be no deductions for minor syntax errors.

1. [**60 points**] Method `containsDuplicates` returns true if the array contains two items with the same state, and false otherwise:

```
public static boolean containsDuplicates( Object [ ] arr )
{
    for( int i = 0; i < arr.length; i++ )
        for( int j = i + 1; j < arr.length; j++ )
            if( arr[ i ].equals( arr[ j ] ) )
                return true;

    return false;
}
```

   (a) What is the Big-Oh running time of `containsDuplicates`?

   (b) If it takes 16 milliseconds to return false to 1000 items, approximately how long would it take to return false for 2000 items?

   (c) Using the Collections API, describe an algorithm (in English, no code) that is more efficient than the one above, and provide the running time of your algorithm, with a brief justification.

2. [**70 points**] *WRITE YOUR SOLUTION ON THE FACING PAGE; IF YOU NEED TWO PAGES, USE THE BACK OF THE LAST PAGE.*

Suppose that a game is represented in the following interface:

```
interface Game
{
    String getWinnerName( );
    String getLoserName( );
    int    winnerPoints( );
    int    loserPoints( );
}
```

Class `GameFactory` contains static method `makeGame` that will return a constructed `Game` object given a `String` representing the game:

```
class GameFactory
{
    public static Game makeGame( String gameInfo )
      { ... }
}
```

You may assume that `makeGame` works as specified.

(a) Implement static method

```
public static Map computeSummary( String [ ] games )
```

Method `computeSummary` accepts an array containing games, in a format suitable for invoking `makeGame`. It returns a `Map` in which the keys are the team names, and the corresponding value is a `List` of `Game` in which the team played. Games in each list should appear in the same order as they appeared in the games parameter.

For instance, suppose `games` contains the following four strings:

```
"A B 60 49"
"A C 51 40"
"C B 45 30"
"A C 52 40"
```

Then in the returned `Map`, we have

```
key            value
A             [ [A B 60 49]  [A C 51 40] [A C 52 40] ]
B             [ [A B 60 49]  [C B 45 30] ]
C             [ [A C 51 40]  [C B 45 30] [A C 52 40] ]
```

In implementing `computeSummary`, you may write any private helpers that you deem necessary (but you must implement them).

(b) Implement static method `mostGames`.

```
public static List mostGames( Map gameSummary )
```

The parameter to `mostGames` is a `gameSummary Map`, in the format shown in part (a). The intent of `mostGames` is to find the team that has played the most games. Since there may be several such teams, `mostGames` returns a `List` of teams who have all played the most games; if there is only one such unique team, the size of the returned `List` is 1. For the above input, a `List` of size 2 containing both teams `A` and `C` is returned.

3. [**70 points**] Suppose that the world is flat, and can be represented as a grid of Boolean variables. Each Boolean variable is true if it represents land, and false if it is water. The grid can be encapsulated in the following interface:

```
interface GridElement
{
    int            getRow( );
    int            getCol( );
    boolean        containsWater( );

    GridElement [] getAdjacent( );   // returns grid elements that are adjacent
                                     // length is < 8 if at edge of world
}
```

Implement method `getWaterArea`:

```
public static int getWaterArea( GridElement pos )
```

`getWaterArea` accepts a `GridElement` as a parameter. If the `GridElement` represents land, `getWaterArea` returns 0. Otherwise it returns the number of `GridElements` that are connected to the `GridElement` parameter via water, including the parameter itself.

For example, in the following grid, in which `W` is water and `L` is land:

```
WWLWLWWLWL
LLLWLWWLLL
LLWLLWLLLL
WLLLLLLWLL
LLLWLLWLWL
```

the results of `getWaterArea` for a few positions are as follows:

GridElement pos	getWaterArea(pos)
(0,0)	2
(0,3)	3
(0,5)	5

In implementing `getWaterArea`, you may create private helpers as you see fit, but you must implement them. You MAY NOT declare any additional static data members.

IMPLEMENT `getWaterArea` *ON THE FACING PAGE.*