COP 3530
Data Structures

Midsemester Exam Version A

Name: ________________________________

June 22, 2004

This exam has 4 questions. Each question starts on a new page. Please answer each question on its 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. [**50 points**] Method `hasSum` returns true if the array contains either one or two different items that sum to parameter `sum` exactly and false otherwise:

```
public static boolean hasSum( int [ ] arr, int sum )
{
    for( int i = 0; i < arr.length; i++ )
    {
        if( arr[ i ] == sum )
            return true;

        for( int j = i + 1; j < arr.length; j++ )
            if( arr[ i ] + arr[ j ] == sum )
                return true;
    }

    return false;
}
```

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

   (b) If it takes 8 milliseconds to return false for 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.

   (d) Does a change to the problem that requires exactly two items to sum (rather than allowing a single item) make the problem easier in a Big-Oh sense? Why/why not?

2. [**50 points**] This question requires that you implement some methods for a class that represents a doubly-linked list. In this question, **both a beginMarker and endMarker are used**; however, there is no field used to keep track of the size. You may assume an appropriate declared nested class `Node`.

   (a) Implement `size` and **PROVIDE ITS BIG-OH RUNNING TIME**.

   ```
   public int size( )
   {

   }
   ```

   (b) Implement the private helper method `remove` in the space shown below:

   ```
   private void remove( Node p )
   {

   }
   ```

   (c) Implement `removeFirst` in the space shown below. You may assume code written by you in the previous part works. You must throw an exception if appropriate.

   ```
   public void removeFirst( )
   {
   ```

3. [**50 points**] Assume that you have a `java.util.Map` in which the keys are names of students (stored as a `String`), and for each student, the value is a `java.util.List` of courses (each course name is a `String`) that the student is enrolled in.

   (a) Write a routine that given a student, returns the list of courses the student is enrolled in.

   (b) Write a routine that given a course, returns the list of students registered for the course.

```
public static List getCourses( Map m, String student )
{








}


public static List getStudents( Map m, String course )
{
```

4. [**50 pts**] Method hasSum returns true if the array contains zero or more items that sum to parameter sum exactly and false otherwise. Example: arr is [3, 4, 9, 1]. If called with sum = 8, the result is true (3,4,1); if called with sum = 0, the result is true (use zero elements); if called with sum = 11, the result is false.

The easiest way to implement hasSum is to use recursion. The public driver and the start of the recursive routine is shown. Implement the recursive routine.

```
public static boolean hasSum( int [ ] arr, int sum )
{
    return hasSum( arr, 0, arr.length - 1, sum );
}

// Returns true if some subset of elements in arr[low..high] adds to sum
// returns false otherwise
// HINT: BASIC IDEA IS TO
//   return true if some a subset of elements in arr[low+1..high] adds to sum OR
//      some subset of elements in arr[low+1...high] adds to sum-arr[low]
public static boolean hasSum( int [ ] arr, int low, int high, int sum )
{
```