NAME (print): _____

Honor Acknowledgement (signature): _____

DO NOT SPEND MORE THAN 10 MINUTES ON ANY QUESTION! If you don't see the solution to a problem right away, move on to another problem and come back to it later.

Before starting, make sure your test contains 18 pages. The last two pages are blank and can be used as scratch paper, but **must be turned in**.

Pages 15 and 16 have the Java classes String, ArrayList, and Scanner

Do **not** discuss this test **with anyone** until the test is handed back.

None of the Java programs or program segments should have syntax errors unless stated. If you think there is a syntax error, then ask. Import statements may not always be shown.

	value	grade
Problem 1	12 pts.	
Problem 2	12 pts.	
Problem 3	10 pts.	
Problem 4	12 pts.	
Problem 5	16 pts.	
Problem 6	32 pts.	
TOTAL:	94 pts.	

PROBLEM 1 : (*What if?:* (12 pts))

Assume that `value` and `other` are declared as type `integer`, and consider the following code segment.

```
if ((value > 7) && (other < 4))
   System.out.print("ok ");
```

For each code segment below, state whether or not it is equivalent to the above code segment. If it is not equivalent, then give input values for `value` and `other` for which the code segments generate different output.

A.

```
if (value > 7)
   System.out.print("ok ");
if (other < 4)
   System.out.print("ok ");
```

B.

```
if ( ! ((value <= 7) && (other >= 4)))
   System.out.print("ok ");
```

C.

```
 if (! ((value <= 7) || (other >= 4)))
    System.out.print("ok ");
```

D.

```
 if (value > 7)
 {
    if (other < 4)
    {
       System.out.print("ok ");
     }
  }
```

PROBLEM 2 : (*Output:* (12 pts))

Consider the following Mystery method. Note that there are preconditions on the arguments passed to the method. First, `number` must be greater than 0 and contain no 0 digits, `value` must be greater than 0 and less than 10.

```
// precondition: number contains no 0's, number > 0
//     0 < value < 10
public int Mystery(int number, int value)
{
   int count = 0;
   int dig = number % 10;
   while (dig > 0)
   {
      if (dig == value)
      {
         count++;
      }
      number = number/10;
      dig = number % 10;
   }
   return count;
}
```

A. What type is the return value for the method Mystery?

B. How many local variables does Mystery have and what are their names?

C. What is the return value for the call Mystery(376646, 6)?

D. What is the return value for the call Mystery(376646, 5)?

E. Describe in words what the method Mystery does.

PROBLEM 3 : (*Need a Place to Stay?:* (10 pts))

Consider the following two classes.

```
public class House {
   private String myAddress;
   private int myNumberOfBedrooms;

   public House(String address, int numberOfBedrooms) {
         // code not shown
   }

   public int getNumberOfBedrooms() {
      // code not shown
   }

   public String getAddress()  {
      // code not shown
   }

   public void setNumberOfBedrooms(int number) {
      // code not shown
   }
}

public class BeachHouse extends House{

   private int myDistanceToBeach; // distance from house to beach

   public BeachHouse(String addr, int numBdrms, int distance) {
      // code not shown
   }

   public String getAddress() {
      // code not shown
   }

   // increase number of bedrooms by 1
   public void additionalBedRoom()
   {
      setNumberOfBedrooms(getNumberOfBedrooms()+1);
   }
}
```

A : Which of the two classes is the superclass?

B : Give the name of a method that is overridden.

C : In the code for the BeachHouse method `additionalBedRoom`, suppose the code is replaced by:

```
myNumberOfBedrooms++;
```

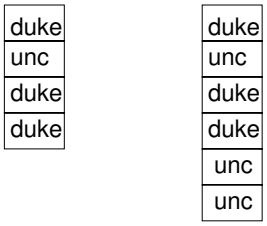
Explain why there is now a compile error.

D : In order for the code change made in part C to compile, what is the best other change within the code to make? Explain.

PROBLEM 4 : (*The equalizer:* (12 pts))

Write the method `equalize` which has one parameter for an ArrayList of Strings. Each entry in the ArrayList is either "duke" or "unc" (there are no other words in the ArrayList). The method `equalize` determines if there are more occurences of one type than the other. If there are, it adds additional strings to the end of the ArrayList so that both "duke" and "unc" occur an equal number of times. Then it returns the ArrayList.

For example, suppose `schools` has the entries shown on the left below. After the statement `schools = equalize(schools);` , then schools has the entries shown on the right below (2 "unc"'s were added at the end of the ArrayList, now there are an equal number of occurences of both "duke" and "unc").



Complete the method `equalize`.

```
public ArrayList<String> equalize(ArrayList<String> schools)
{









}
```

PROBLEM 5 : (*No More Spam* (16 pts))

PART A *(8 pts)*:

Write the method `replaceAt` whose header is given below. This method has one String parameter. It returns this String with all occurences of "@" replaced with " AT " (note the two blanks, one before A and one after T).

For example `replaceAt("rodger@cs.duke.edu")` returns `"rodger AT cs.duke.edu"`, `replaceAt("@a@b@c")` returns `" AT a AT b AT c"`, and `replaceAt("go duke go")` returns `"go duke go"` (unchanged since there are no @'s.).

Complete the method `replaceAt` below.

```
// returns a String equivalent to phrase with all occurrences of
//   "@" replaced with " AT "
public String replaceAt(String phrase)
{




}
```

PART B ***(8 pts)***:

Write the method `replaceAllAts` whose header is given below. This method has one Scanner parameter named `input` that is already bound to a file for reading. It returns an ArrayList where each entry is a line from the file with all occurences of "@" replaced with " AT " (note the two blanks, one before A and one after T).

For example, if the file bound to the Scanner input is shown on the left below, then after the statement `data = replaceAllAts(input);` is executed (where data is an ArrayList of type String) the ArrayList data is shown below on the right.

```
rodger@cs.duke.edu
a@duke.edu b@duke.edu
@@@@
go duke go
no more data
```

rodger AT cs.duke.edu
a AT duke.edu b AT duke.edu
AT AT AT AT
go duke go
no more data

In writing `replaceAllAts` you should call the method `replaceAt` that you wrote in Part A. Assume that `replaceAt` is correct, regardless of what you wrote.

Complete the method `replaceAllAts` below.

```
public ArrayList<String> replaceAllAts(Scanner input)
{




}
```

PROBLEM 6 : (*When are we going to get snow?* (32 pts))

PART A *(14 pts)*:

The class Skier shown below represents a Skier who races and saves the following state representing the Skier: the skier's name, the skier's number (each skier has a different number they wear), and the skier's best time so far.

A Skier will race down a run several times and only store the fastest run time. If the skier hasn't completed a run yet, the time stored is -1.

```
public class Skier {

   private String myName; // name of skier
   private int myNumber;  // unique number of skier
   private int myTime;    // best time in seconds
                          // best time is -1 if skier hasn't raced yet

   public Skier(String name, int number) {   // code not shown
   }

   // returns name of skier
   public String getName() {    // code not shown
   }

   // returns number of skier
   public int getNumber()   {    // code not shown
   }

   // returns best race time of skier so far, or a negative number if the
   //       skier has not raced yet.
   public int getTime() {     // code not shown
   }

   // sets skiers best time to raceTime if this is the first run
   // or if raceTime is faster than the skiers best time previously
   public void setTime(int raceTime) {
        // code not shown
   }

   // not all methods shown
}
```

PART A1 ***(4 pts)***:

Complete the Skier constructor shown below. It sets the Skier's time to -1 and uses the parameters to initialize the rest of the state.

```
// Constructor
public Skier(String name, int number)
{









 }
```

PART A2 ***(6 pts)***:

Complete the following Accessor methods to return the state data.

```
// returns name of skier
public String getName() {



}

// returns number of skier
public int getNumber()   {



}

// returns best race time of skier so far, or -1 if the
//       skier has not raced yet.
public int getTime() {



}
```

PART A3 ***(4 pts)***:

Complete the method `setTime` to set the skiers best time to raceTime if this is the first run (remember that times are initialized to -1 if the skier hasn't raced yet). Otherwise, set the skiers time to raceTime only if it is a faster time than the skiers previous runs.

```
// sets skiers best time to raceTime if this is the first run
// or if raceTime is faster than the skiers best time previously
public void setTime(int raceTime)
{









}
```

PART B *(18 pts)*:

The class `SkiRace` keeps track of lots of Skiers. Part of its class is shown below.

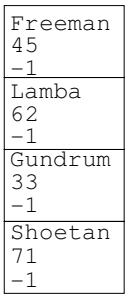
```
public class SkiRace {

   // state
   private ArrayList<Skier> mySkiers;

   // constructor - reads in skiers names and race numbers
   public SkiRace(Scanner input)  {
       // code not shown
   }

   // For each skier completes one round of skiing
   public void skiRound()    {
       // code not shown
   }

   // Print the name of the skier with the best time. If there
   // is a tie, then print all such names with that time.
   public void PrintNamesWithBestSkiTimes()    {
        // code not shown
   }

   // Print the name, number and best times of all skiers.
   // If a skier has not skied yet, then print 0 for their time.
   public void printResults()    {
        // code not shown
   }
}
```

PART B1 ***(8 pts)***:

Write the constructor for the `SkiRace` class whose header is given below. The constructor has one parameter, a Scanner that is already bound to a file. It creates and initializes the ArrayList `mySkiers` by reading in names (each name is one word) and the numbers of skiers.

For example, consider the data file shown on the left below. After a SkiRace object is constructed with the Scanner bound to this file, the ArrayList of Skiers is shown on the right below.

```
Freeman 45
Lamba 62
Gundrum 33
Shoetan 71
```



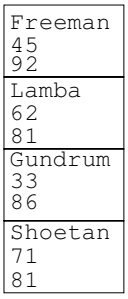
Assume you do not know how many Skiers are in the data file. Complete the constructor`SkiRace` whose header is shown below.

```
// constructor - reads in skiers names and race numbers
public SkiRace(Scanner input)
{




}
```

PART B2 ***(10 pts)***:

Write the method `PrintNamesWithBestSkiTimes` whose header is given below. This method assumes the skiers have already done at least one run or more (so there are no negative times in the ArrayList), and it prints the name of the skier with the fastest time. If more than one skier has the fastest time, it prints all such names. You may assume that all times in the ArrayList are between 0 and 1000.

For example, suppose several runs have been completed and `mySkiers` contains the items shown below (the last number in each Skier entry in the ArrayList is the time). Then the output as a result of the call to PrintNamesWithBestSkiTimes would be: Lambda ShoeTan. (they both have the fastest time of 81 seconds).




```
// Print the name of the skier with the fastest time. If there
// is a tie, then print all such names with that time.
// Assume all times are between 0 and 1000.
public void PrintNamesWithBestSkiTimes()
{









}
```

```
public class String
{
    // Returns the length of this string.
    public int length ()

    // Returns a new string that is a substring
    // of this string. The substring begins at
    // the specified beginIndex and extends to
    // the character at index endIndex - 1.
    public String substring (int beginIndex,
                             int endIndex)

    // Returns the index within this string of
    // the first occurrence of str
    // returns -1 if str is not found
    public int indexOf (String str)

    // Returns the index within this string of the
    // first occurrence of str after index start
    // returns -1 if str is not found
    public int indexOf (String str, int start)
}

public class ArrayList
{
    // Constructs an empty list
    public ArrayList ()

    // Returns the number of elements in this list.
    public int size ()

    // Returns element at index in this list.
    public Object get (int index)

    // Replaces the element at the specified position
    // in this list with the specified element.
    public Object set (int index, Object element)

    // Appends specified element to end of this list.
    public boolean add (Object o)
}
```

```
public class Scanner
{
    // Create Scanner that reads data from a file.
    public Scanner (File file)

    // Create Scanner that reads data from a string.
    public Scanner (String str)

    // Change delimiters used to separate items
    public void useDelimiter (String characters)

    // Check if more items are available
    public boolean hasNext ()

    // Get next delimited item as a string
    public String next ()

    // Get next line as a string
    public String nextLine ()

    // Get next delimited item as an integer value
    public int nextInt ()

    // Get next delimited item as a Double value
    public int nextDouble ()
}
```