# CS 312 – Final – Fall 2013

Your Name SOLUTION SOLUTION SOLUTION SOLUTION SOLUTION

Your UTEID SOLUTION SOLUTION SOLUTION SOLUTION SOLUTION _

| Problem Number | Topic | Points Possible | Points Off |
|---|---|---|---|
| 1 | short answer 1 | 12 | |
| 2 | program logic | 16 | |
| 3 | short answer 2 | 14 | |
| 4 | return methods | 10 | |
| 5 | arrays 1 | 20 | |
| 6 | critters | 20 | |
| 7 | arrays 2 | 20 | |
| 8 | file processing | 15 | |
| 9 | 2d arrays | 15 | |
| | | TOTAL POINTS OFF: | |
| | | SCORE OUT OF 142: | |

Instructions:
1. Please turn off your cell phones
2. You have 3 hours to complete the test.
3. You may not use a calculator.
4. Please make your answers legible.
5. When code is required, write Java code.
6. You may break problems up into smaller methods. (In other words you can add helper methods.)
7. Style is not evaluated when grading.
8. The proctors will not answer questions. If you believe there is an error or a question is ambiguous, state your assumptions and answer based on those assumptions.
9. When you finish, show the proctor your UTID, turn in the exam and all scratch paper.

**1. Short Answer 1 - Expressions. 1 point each, 12 points total.**

For each Java expression in the left hand column, indicate the resulting value in the right hand column. **You must show a value of the appropriate type. For example, 7.0 rather than 7 for a double and "7" instead of 7 for a String. Answers that do not indicate the data type correctly are wrong.**

| | | |
|---|---|---|
| A. | `3 * 4 + 18 / 6` | **15** |
| B. | `14 / 5` | **2** |
| C. | `2.0 * 3 / 4` | **1.5** |
| D. | `(int) (1.8 + .15)` | **1** |
| E. | `15 % 8` | **7** |
| F. | `2 + 4 + "CS" + 3 + 1` | **"6CS31"  (no quotes = -1)** |
| G. | `10005 / 10 / 100 * 3` | **30** |
| H. | `"CS312-2013".substring(6)` | **"2013" (no quotes = -1)** |

x is an int variable initialized previously

| | | |
|---|---|---|
| I. | `x == 12 && x == 10` | **false** |
| J. | `Math.ceil(1.01 * 10)` | **11.0  (no .0 = -1)** |
| K. | `3 % 5 == 0 \|\| 12 % 2 == 0` | **true** |
| L. | `!(12 < 18 && 12 > 5)` | **false** |

**2. Program Logic - 16 points.** Consider the following method. For each of the four points labeled by comments and each of the assertions in the table, write whether the assertion is ***always*** true, ***sometimes*** true, or ***never*** true at that point in the code. Abbreviate ***always*** with an A, ***sometimes*** with an S and ***never*** with an N.

```
public static int assertionPracticeFinal(int value){
     if(value < 2)
          return -1;
     else {
          int result = 0;
          int i = 2;
          // POINT A
          while(i < value && result < value) {
               // POINT B
               int temp = value % i;
               if(temp == 0) {
                    result += i;
                    // POINT C
               }
               i++;
          }
          // POINT D
          return result;
     }
}
```

Abbreviate ***always*** with an A, ***sometimes*** with an S and ***never*** with an N.

| | `result == 0` | `i < result` | `result < value` | `value % i == 0` |
|---|---|---|---|---|
| Point A | **A** | **N** | **A** | **S** |
| Point B | **S** | **S** | **A** | **S** |
| Point C | **N** | **S** | **S** | **A** |
| POINT D | **S** | **S** | **S** | **S** |

**3. Short Answer 2 - 1 point each, 14 points total.** For each code snippet state the exact output to the screen. Assume all necessary imports have been made.
If the snippet contains a syntax error or other compile error, answer COMPILE ERROR.
If the snippet results in a runtime error or exception, answer RUNTIME ERROR.

```
A.                                   5 3
int x = 3;
ma(x);
System.out.print(x);

public static void ma(int x) {
     x += 2;
     System.out.print(x + " ");
}
```

```
B.
int[] datab = {5, 3, 7};                       [5, 4, 2]
mb(datab);
System.out.println(Arrays.toString(datab));

public static void mb(int[] ds) {
     ds[2] -= ds[0];
     ds[1]++;
     ds = new int[2];
     ds[1] = 12;
}
```

```
C.                                                       [C, D, F]
ArrayList<String> listc = new ArrayList<String>();
listc.add("C");
listc.add("E");
listc.add("F");
listc.add(1, "D");
listc.remove(2);
System.out.println(listc.toString());
```

```
D.                                        18
System.out.print(recMystery(4));

public static int recMystery(int x) {
     if(x <= 0)
          return -2;
     return x * 2 + recMystery(x - 1);
}
```

For the rest of the short answer questions consider these classes:

```
public class Media {
     public int size() { return -1; }
     public String toString() {return "Media"; }
}

public class Video extends Media {
     public String toString() { return "Video"; }
}

public class Film extends Video {
     public int size() { return 1000; }
     public String toString() { return "Video"; }
}

public class Printed extends Media {
     public int size() { return 200; }
     public String toString() {return "Printed"; }
}

public class Book extends Printed {

     private int pages;

     public Book(int p) { pages = p; }
     public int size() { return pages; }
     public boolean hardcover() { return pages >= 300; }
     public void changePages() { pages += 100; }
}
```

```
E.                              -1
Video v1 = new Video();
System.out.print(v1.size());
```

```
F.                              false false
Book b1 = new Book(400);
Book b2 = new Book(400);
System.out.print((b1 == b2) + " " + b2.equals(b1));
```

```
G.                              200
Media m1 = new Printed();
System.out.print(m1.size());
```

```
H.                             COMPILE ERROR
Media m2 = new Book(100);
System.out.print(m2.hardcover());
```

```
I.                             COMPILE ERROR
Media m3 = new Media(300);
System.out.print(m3.size());
```

```
J.                             true
Book b3 = new Book(500);
System.out.print(b3.hardcover());
```

```
K.                             COMPILE ERROR
Book b4 = new Book();
System.out.print(b4.size());
```

```
L.                                     300
Book b5 = new Book(100);
mk(b5);
System.out.print(b5.size());

public static void mk(Book b5) {
     b5.changePages();
     b5.changePages();
}
```

```
M.                             1149

Media[] mList = {new Media(), new Film(), new Book(150)};
int total = 0;
for(Media m : mList)
     total += m.size();
System.out.print(total);
```

```
N.                             COMPILE ERROR
Book b6 = new Book(100);
b6.pages += 400;
System.out.println(b6.hardcover());
```

**4. Return methods - 10 Points.** Write the method `pythagoreanTriple`. The method accepts 3 `int` parameters. The method returns `true` if the three `int`s form a Pythagorean Triple, `false` otherwise.

A Pythagorean Triple consists of three positive ints a, b, and c such that $a^2 + b^2 = c^2$

Examples:

```
pythagoreanTriple(5, 3, 4) -> returns true
pythagoreanTriple(4, 5, 3) -> returns true
pythagoreanTriple(2, 2, 4) -> returns false
pythagoreanTriple(5, 12, 13) -> returns true
pythagoreanTriple(10, 8, 6) -> returns true
pythagoreanTriple(8, 7, 6) -> returns false
pythagoreanTriple(5, 5, 5) -> returns false
```

You may not use any other Java classes or objects. Specifically you may **not** use the `Math` class.

As you can see from the examples the parameters are not in any particular order.

```
    public static boolean pythagoreanTriple(int a, int b, int c) {
        int aSquared = a * a;
        int bSquared = b * b;
        int cSquared = c * c;
        return aSquared + bSquared == cSquared ||
                aSquared + cSquared == bSquared ||
                bSquared + cSquared == aSquared;
    }
```

3 points: calculate $a^2$, $b^2$, and $c^2$ (-3 if use Math class)
3 points: all 3 cases checked: (1 point each)
3 points: proper logic with || (or returns)
1 point: return value

**5. Arrays 1 - 20 points.** Write the method `elementsMatch`. The method has two parameters. Both parameters are arrays of `String`s equal in length. The method creates and returns an array of `boolean`s equal in length to the arrays of `String`s sent as parameters.

Each element in the returned array is `true` if the corresponding elements from the array of `String`s *match*. For this question two `String`s match if either of the following conditions are met:

- the `String`s are the same length
- the first character in the first `String` equals the first character in the second `String` AND the last character in the first `String` equals the last character in the second `String`

Examples of `String`s that match according to these criteria:

`"AAA"` and `"BBB"` `->` same length
`"BAT"` and `"BIT"` `->` first characters and last characters equal, B = B, T = T
`"BEAT"` and `"BAT"` `->` first characters and last characters equal, B = B, T = T
`"#"` and `"#display#"` `->` first and last characters equal, # = # and # = # (Given a `String` of length 1 the single character is the first AND last character.)

`""` and `""` `->` same length, 0

Note, `"A"` and `""` do not match for this question. The empty String does not have a first or last character.

Another example showing two arrays and the result:

```
 0          1        2       3      4      5          6
{"ABBBBB", "ADBC",  "",     "DAD", "MOM", "DAD",    "&*AA"}
{"ABBA",   "AC",    "A",    "DOS", "M",   "DRIED", "&+A"}

{false,    true,    false, true, true,  true,     true}
```

Note, the elements at index 3 match because they are the same length.

You may not use any other methods or classes in your answer other than Java's built in native arrays, and methods from the `Math` class and `String`s. You may use the `String` `charAt` and `length` methods.

Assume neither array sent as a parameter is null. Assume none of the elements in either array are `null`.

The method does not alter the parameters s1 or s2 in any way.

Complete this method on the next page:

```
public static boolean[] elementsMatch(String[] s1, String[] s2) {
```

# Complete this method on the next page.

```
// Neither s1 or s2 is null. No elements of s1 or s2 are null.
// s1.length == s2.length
    public static boolean[] elementsMatch(String[] s1, String[] s2) {
        boolean[] result = new boolean[s1.length];
        for(int i = 0; i < result.length; i++) {
            String str1 = s1[i];
            String str2 = s2[i];
            boolean sameLength = str1.length() == str2.length();
            boolean firstLastCharMatch = false;
            if(str1.length() > 0 && str2.length() > 0) {
                firstLastCharMatch = str1.charAt(0) == str2.charAt(0)
                    &&  str1.charAt(str1.length() - 1) ==
                               str2.charAt(str2.length() -1);
            }
            result[i] = sameLength || firstLastCharMatch;
        }
        return result;
    }
```

2 points: create resulting array correctly
3 points: for loop correct (-1 if OBOE, okay to take off 1, 2 or 3)
2 points: access elements in array correctly
3 points: check same length correctly
3 points: check first char same correctly
3 points: check last char same correctly
3 points: set value in result correctly if either condition met
1 point: return result

**6. Critters - 20 Points.** Implement a complete Bear class from the Critters assignment.

| | |
|---|---|
| **constructor** | `public Bear(int turnsHibernate)` |
| **fight behavior** | if currently hibernating `ROAR`, otherwise `SCRATCH` |
| **color** | `Color.BLACK` |
| **movement behavior** | if a Bear is hibernating (see below) the Bear stays put when asked to move. In other words it returns `CENTER`.<br><br>If a Bear is not hibernating it always moves `NORTH`. |
| **eating behavior** | If a Bear is not hibernating return `false`, otherwise return `true` |
| **toString** | `"H"` if hibernating, `"B"` if not hibernating |

Bears are either in a hibernating or not hibernating state. When created Bears are not hibernating. A bear starts to hibernate after it eats. The int to the constructor indicates the number of moves a Bear hibernates after eating. After the number of turns has elapsed the Bear is no longer hibernating.

Calls to the `getMove` method are the only way the number of turns a Bear has hibernated is incremented.

Note, hibernation time is in addition to any time a Bear spends sleeping after eating. Sleeping after eating is handled by the simulation controller. You must implement the hibernation behavior in your Bear class. Abstractly, hibernation time is similar to a continuation of sleeping, but the Bear can fight and mate.

Recall the Critter class:
```
public abstract class Critter {

     public boolean eat() {  return false; }

     public Attack fight(String opponent) {
          return Attack.FORFEIT;
     }

     public Color getColor() { return Color.BLACK; }

     public Direction getMove() { return Direction.CENTER; }

     public String toString() { return "?"; }

     // constants for directions
     public static enum Direction {
          NORTH, SOUTH, EAST, WEST, CENTER
     };

     // constants for fighting
     public static enum Attack {
          ROAR, POUNCE, SCRATCH, FORFEIT
     };
```

Write the complete Bear class including the class header and instance variables following the specification given on the previous page.

```
public class Bear extends Critter{

    private int turnsToHibernate;
    private int turnsSoFar;
    private boolean hibernating;

    public Bear(int turns) {
        turnsToHibernate = turns;
    }

    public boolean eat() {
         boolean result = hibernating;
         hibernating = true;
        return result;
    }

    public Attack fight(String opponent) {
        return hibernating ? Attack.ROAR : Attack.SCRATCH;
    }

    public Direction getMove() {
        Direction result
                = hibernating ? Direction.CENTER : Direction.NORTH;
        if(hibernating) {
            turnsSoFar++;
            if(turnsSoFar == turnsToHibernate) {
                hibernating = false;
                turnsSoFar = 0;
            }
        }
        return result;
    }

    public String toString() {
        return hibernating ? "H" : "B";
    }
}
```

2 points: method header, with extends Critter
2 points: instance vars for: turns to hibernate and turns hibernated so far (boolean optional)
2 points: constructor correct
3 points: eat method correct
2: points: fight method correct
2 points toString correct
7 points: getMove correct (result correct result: 2 points, if hibernating update turns hibernating: 2 points,
check if hibernation over and reset variables: 2 points)

**7. Arrays 2 - 20 Points**. Given an array of `Point` objects, write a method `getMinDistance` that finds the minimum distance between two distinct points. Do not compare a given Point to itself, but it is possible for 2 or more Points in the array to equal each other resulting in a minimum distance of 0.0.

The public portion `Point` class for this method:

```
public class Point {
      public Point(int x, int y)
      public int getX()
      public int getY()

      // returns distance between 2 Points
      public double getDistance(Point otherPoint)
}
```

In the example `Point` objects are shown in the form [x, y].

`{[1, 12], [3, 5], [5, 3], [8, 12]} returns 2.8284271247461903` (square root of 8)

`{[12, -5], [0, 0]} returns 13.0`

`{[1, 2], [3, 5], [5, 3], [8, 12], [2, 1], [5, 3]} returns 0.0`

You may use Java native arrays, the given `Point` class and the `Math` class.

Assume the array `length` is greater than or equal to 2 and that no elements are `null`.

The method does not alter the array of `Point` objects in any way.

# Complete the method on the next page.

```
// pts.length >= 2, no elements of pts are null
    public static double getMinDistance(Point[] pts) {
        double min = pts[0].getDistance(pts[1]);
        for(int i = 0; i < pts.length; i++)
            for(int j = i + 1; j < pts.length; j++) {
                double distance = pts[i].getDistance(pts[j]);
                if(distance < min)
                    min = distance;
            }
        return min;
    }
```

2 points : initialize min correctly (arbitrary value **not** okay)

3 points: outer loop correct

3 points: inner loop correct (or if to not check Point with itself)

4 points: access Point objects from array and call methods 2 points

3 points: calculate distance (method or long way)

4 points: check if current distance smallest and if so, set result

1 point: return result

**8. File Processing - 15 Points.** Write a method named `plusScores` that accepts as a parameter a `Scanner` connected to a file containing a series of lines representing student records. Each student record consists of 2 lines. The first line has the student's name and the second line has a series of plus and minus characters. Below is a sample file:

```
Ryan
--+-+
Chris
++-+
Mike D
+++++++
Edaena
++-++-+-
Leif Johnson
-------------
Mubashir Adnan Q
++-++++
```

The number of plus/minus characters varies, but you may assume that at least one such character appears in the second line of each record and that no characters other than plusses and minuses appear on the second line of each record.

For each student print out the student's name and percentage of plusses using the format shown below.

For example, if the input above is stored in a `Scanner` called `input`, the call of `plusScores(input)`; produces the following output:

```
Ryan: 40.0% plus
Chris: 75.0% plus
Mike D: 100.0% plus
Edaena: 62.5% plus
Leif Johnson: 0.0% plus
Mubashir Adnan Q: 85.71428571428571% plus
```

You may use any methods from the Java `Scanner` and `String` classes. You may not use any other Java classes or objects.

# Complete the method on the next page.

```
// input != null, input already connected to File
    public static void plusScores(Scanner input) {
        while(input.hasNextLine()) {
            System.out.print(input.nextLine() + ": ");
            String data = input.nextLine();
            int numPlusses = 0;
            for(int i = 0; i < data.length(); i++)
                if(data.charAt(i) == '+')
                    numPlusses++;
            System.out.println(100.0 * numPlusses / data.length()
                         + "% plus");
        }
    }
```

3 points: while loop correct

1 point: print out name (can use variable)

1 point read in line of input

2 points: loop through characters in data

2 point: check if char a plus

2 points: track number of plusses

3: points: calculate % correctly (-1 if int division, -1 if not * 100)

1 point: print out % data and plus

**9. 2d Arrays - 15 points.** Write a method `specialSum` that returns the sum of the elements in a 2D array of `int`s with the following special conditions:

- Any negative element that occurs in a row with an even index results in the element * 2 being added to the total sum. For example if an element in row 2 equals -4 then -8 (-4 * 2) is added to the sum instead of -4. (Note, 0 is an even number.)
- Any negative element that occurs in a row with an odd index results in element * 5 being added to the total sum. For example if an element in row 7 equals -8 then -40 (-8 * 5) is added to the sum instead of -8.

Example. Given the following 2d array:

| | | | | |
|---|---|---|---|---|
| 2 | -1 | 0 | -2 | row 0, result of row = -4 |
| 3 | -1 | 1 | -4 | row 1, result of row = -21 |
| 12 | 0 | -1 | -2 | row 2, result of row = 6 |

Overall result = -19

Complete the following method. You may only use Java's native 2d arrays. Assume `mat` is not `null` and that `mat` is a rectangular matrix. (All rows have the same number of columns.). The method does not alter the elements of `mat`.

```java
public static int specialSum(int[][] mat) {
    int total = 0;
    for(int r = 0; r < mat.length; r++)
        for(int c = 0; c < mat[r].length; c++) {
            int val = mat[r][c];
            if(val > 0)
                total += val;
            else if(r % 2 == 0)
                total += val * 2;
            else
                total += val * 5;
        }
    return total;
}
```

2 points: total var and init to 0

3 points: outer loop correct

3 points inner loop correct

2 points: regular case

2 points: negative value in even row case

2 points: negative value in odd row case
1 point: return result

**Extra Credit - 1 point each, 3 points total:**

**In class on the Wednesday before Thanksgiving we had a lively discussion regarding some of the research done by professors in the CS department.**

**1. Who is the chairman of the UT Computer Science Department?**

**<u>BRUCE PORTER OR PORTER</u>**

**2. What is the chairman's research area within Computer Science?**

**<u>ARTIFICIAL INTELLIGENCE OR AI</u>**

**3. The chairman, along with their graduate students, wrote a program to take an Advanced Placement exam. What was the subject area of the exam?**

**<u>CHEMISTRY</u>**

**FYI the test taking program scored a 3. It was part of a research contest named Project Halo funded by Vulcan Inc., which in turn is funded by Paul Allen, one of the founders of Microsoft.**