Your Name_____

Your UTEID _____

Problem Number	Topic	Points Possible	Points Off
1	expressions	10	
2	program logic	20	
3	code tracing	15	
4	critters	15	
5	arrays	15	
6	strings	15	
7	2d arrays	15	
8	ArrayList	10	
9	file processing	20	
	TOTAL POINTS OFF:		
	SCORE OUT OF 135:		

Instructions:

1. Please turn off your cell phones
2. You have 3 hours to complete the test.
3. You may not use a calculator or any other electronic device.
4. When code is required, write Java code.
5. Ensure you follow the restrictions of the question.
6. You **are** allowed to add helper methods. (break problem up)
7. When you finish, show the proctor your UTID, turn in the exam and all scratch paper.

1. Short Answer 1 - Expressions. 1 point each, 10 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, "7" instead of 7 for a String, and '7' for a char.

Answers that do not indicate the data type correctly are wrong.

A. `1.5 + 10 / 4 + 5 / 7 * 2.0` _____

B. `42 % 38 + 38 % 37 + 7292 / 100 % 10` _____

C. `2 + 3 + "dog" + 4 + 5` _____

D. `((double) 4) / 10 + 5 / 4` _____

E. `(int) (3.999 * 10.0)` _____

F. `(3 * 5 > 10) || (10 / (6 - 2 * 3) > 0)` _____

G. `(3 <= 3 * 3) + " " + (2 * 2 != 4)` _____

H. `"ios_swift".substring(2,6).substring(2)` _____

I. `"cat" == "dogcat".substring(3)` _____

J. `! (3 < 2 * 4 || 10 * 10 == 8 * 8)` _____

2. Program Logic - 20 points. Consider the following method. For each of the five 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 assertionPractice(int num) {
     int result = 0;
     boolean half = false;
     int temp = 0;
     // POINT A
      if (num > 2) {
          while (temp < num) {
               // POINT B
               half = Math.random() < 0.5;
               result++;
               if (half) {
                    // POINT C
                    temp++;
               } else {
                    temp = 0;
               }
               // POINT D
          }
          // POINT E
     }
     return result;
}
```

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

	temp < num	result <= temp	temp == 0	half == true
Point A				
Point B				
Point C				
POINT D				
POINT E				

3. Code Tracing - 1 point each, 15 points total. For each code snippet state the exact output to the screen. **Placed your answers to the right of the snippet and put a box around each answer.**
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.
If the snippet results in an infinite loop, answer INFINITE LOOP.

A.
```
int x1 = 5;
int y1 = 3;
a(x1, y1);
System.out.print(x1 + " " + y1);

public static void a(int x1, int y1) {
     x1 += y1;
     y1 = 4;
     System.out.print(x1 + " " + y1 + " ");
     y1 += 2;
}
```

B.
```
double[] d2 = {2.5, 3, 1.0};
int b2 = 2;
b(d2, b2);
System.out.print(Arrays.toString(d2));

public static void b(double[] vals, double a) {
     vals[1] += vals.length;
     vals[0] += a;
     vals[2]++;
}
```

C.
```
int[] data3= new int[2];
data3[1] = 7;
c(data3);
System.out.print(Arrays.toString(data3));

public static void c(int[] data) {
     data[0] += 3;
     data = new int[3];
     data[1] += 5;
     System.out.print(Arrays.toString(data) + " ");

}
```

D.

```
String[] names4 = {"KK", "", "G", "C"};
d(names4);
System.out.print(Arrays.toString(names4));

public static void d(String[] strs) {
     strs[0] = strs[2] + strs[3];
     strs[1] += strs.length;
     strs[2] = "B";
}
```

E.

```
String[] strs5 = new String[5];
System.out.print(strs5[3].length() + " " + strs5.length);
```

F.

```
int[][] mat6 = new int[3][5];
System.out.print(mat6[1].length + " " + mat6[2][4]);
```

G.

```
String s7 = "0.5    7    2.5       K    3.4    BB      5";
Scanner sc7 = new Scanner(s7);
double t7 = 0.0;
while (sc7.hasNextDouble()) {
     t7 += sc7.nextDouble();
}
System.out.print(t7 + " " + sc7.hasNext());
```

H.

```
ArrayList<String> list = new ArrayList<String>();
list.add("A");
list.add(0, "B");
list.add("C");
list.add(0, "D");
list.add(list.get(1));
System.out.print(list);
```

I. Assume we are implementing two classes, `Page` (as in a page in a book) and `Book`. Assume the `Page` class is already done and has methods to get the number of words on the page and a method to get the number of images on the page. We want to implement the `Book` class and want methods to get the number of words in the book and the number of images in the book. Would it be a **good, logical design** to have the `Book` class extend the `Page` class? Yes or No. Circle your answer:

YES NO

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

```
public class WirelessDevice {
     private int cost;
     private int dataRate;

     public WirelessDevice(int cost) {
          this.cost = cost;
          dataRate = cost * 3;
     }

     public WirelessDevice() {
          cost = 2;
          dataRate = 4;
     }

     public String toString() { return "WD" + getTotal(); }
     public int getTotal() { return cost + dataRate; };
     public int getCost() { return cost; };
}

public class Phone extends WirelessDevice {

     public int getCost() {   return 12; }
}

public class Tablet extends WirelessDevice {

     public Tablet() {
          super(5);
     }

     public int getTotal() {  return 15;     }

     public String getLabel() { return "TT"; }
}

public class TwoInOne extends Tablet {

     public String toString() { return "TI" + getCost(); }
}
```

J. Consider the following statements. For each, circle if it is legal or if it causes a syntax error.

```
TwoInOne tio  = new Tablet();       // legal       syntax error

WirelessDevice wd1 = new TwoInOne();// legal       syntax error
```

K. Consider the following statements. For each, answer circle it is legal or if it causes a syntax error.

```
Phone p1 = new TwoInOne();  // legal       syntax error

Object obj =  new Phone();  // legal       syntax error
```

For each code snippet state the exact output to the screen.

L.
```
TwoInOne t2 = new TwoInOne();
System.out.print(t2.toString() + " " + t2.getTotal());
```

M.
```
WirelessDevice wd2 = new Tablet();
System.out.print(wd2.getCost() + " " + wd2.getTotal());
```

N.
```
Tablet t3 = new Tablet();
System.out.print(t3 + " " + t3.getLabel());
```

O. The following class has 2 compile errors. What are they?

```
public class Wearable extends WirelessDevice {

     public int screens;

     public Wearable(int s) {
          super(4, 5);
          screens = s;
     }

     public int cost() {
          return 100;
     }

     public void upgrade() {
          dataRate += screens * 10;
     }
}
```

What are the two compile errors in the `Wearable` class?

1.

2.

4. Critters - 15 Points. Implement a complete `Sidewinder` class from the Critters assignment.

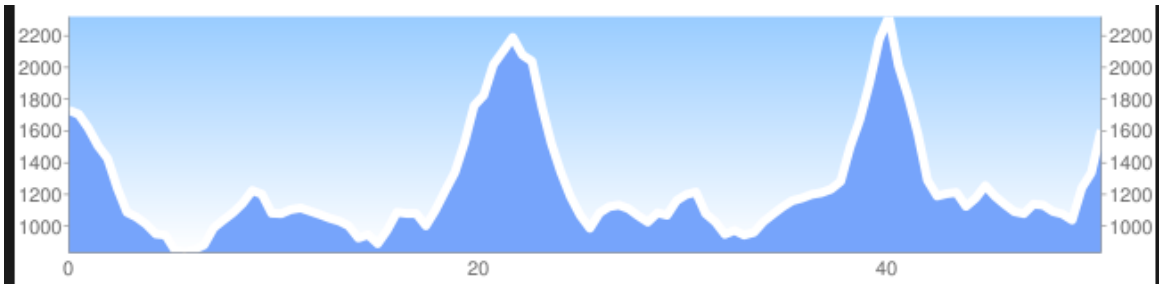
constructor	`public Sidewinder(int addToLeg)`
fight behavior	`ROAR` if the opponent looks like an Ant Critter, "%", otherwise `SCRATCH`.
color	Always returns `Color.GRAY`.
movement behavior	A Sidewinder alternates between moving North and West. The sidewinder moves a certain number of times in one direction before changing directions. The number of steps in a direction starts at 1 but then increases by the amount sent to the Sidewinder constructor. So for exam if the addToLeg value for a Sidewinder was 2, then the Sidewinder would move in the following way: NORTH, WEST, WEST, WEST, NORTH, NORTH, NORTH, NORTH, NORTH, WEST, ... 1 step NORTH, 3 steps WEST, 5 steps NORTH, 7 steps WEST, and so forth.
eating behavior	Returns `true` if total number of steps the sidewinder will take on the current leg is greater than 20, false otherwise.
toString	Always returns "~", a tilde.

```
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 `Sidewinder` class based on the above specification on the next page.
Include the class header and instance variables. Assume any necessary imports are already made.

You may use the Critter and Color classes, and the Attack and Direction enumerated types, but no other built in Java classes or methods.

5. Arrays - 15 Points Complete a method `difficultTrail`. The method has three parameters, an array of `int`s named `elevations`, an `int` named `minChange` representing the minimum change between two elements to qualify for a difficult segment, and an `int` named `minSegments` for the minimum number of difficult segments necessary for the trail to be classified as a difficult trail.



The array of ints sent to the `difficultTrail` method represents the elevations of a walking trail at spots along the trail. Each element represents the elevation at a marker. The markers occur every 100 feet in length. So a steeply rising trail might look something like this:

```
[50, 61, 73, 83, 95, 100, 110, 90, 125, 140, 140]
```

The elevation of the first trail marker is at an elevation of 50 feet. The second trail marker, 100 feet down the trail, is at an elevation of 61 feet, indicating the trail rose 11 feet from the first marker to the second. It is possible for the trail to go down (descend) as moving from left to right.

The method returns true if there are at least `minSegments` in the trail represented by the array of `int`s. with elevation of changes of greater than or equal to `minChange` when walking along the trail from the start of the array to the end of the array, left to right.

Examples of expected results for calls to
`difficultTrail(int[] elevations, int minChange, int minSegments)`

```
difficultTrail({50, 60, 75, 95}, 10, 3) -> returns true

difficultTrail({50, 60, 75, 95}, 15, 3) -> returns false

difficultTrail({50, 60, 75, 95}, 15, 2) -> returns true

difficultTrail({50, 60, 75, 95}, 5, 3) -> returns true

difficultTrail({50, 60, 75, 95}, 5, 5) -> returns false

difficultTrail({50, 60, 75, 95, 50}, 25, 1) -> returns false

difficultTrail({50, 60, 75, 95}, 20, 1) -> returns true
```

You may not use any other built in Java class or methods except native arrays. Of course you may use the `length` field of arrays.
Your solution will be graded on efficiency as well as correctness.

```
// assume elevations != null, minChange > 0, minSegments > 0
public static boolean difficultTrail(int[] elevations,
                                     int minChange, int minSegments) {
```

6. Strings - 15 Points. Write a method called `acronym` that given a `String` parameter containing a phrase, creates and returns an acronym of the phrase. For example, the following call:

```
acronym("self-contained underwater breathing apparatus")
```

shall return `"SCUBA"`. The acronym is formed by combining the capitalized first letters of each word in the phrase.

Words in the phrase will be separated by some combination of spaces and dashes. There might be extra spaces or dashes at the beginning or end of the phrase.

The String will only contain English letters, spaces, and dashes.
The String is guaranteed to contain at least one word.

Below are several examples calls.

`acronym(" Texas ")` returns `"T"`

`acronym("WhatALONGwordThisIs")` returns `"W"`

`acronym("university-- ---     texas")` returns `"UT"`

`acronym("advanced  placement")` returns `"AP"`

`acronym("a--p")` returns `"AP"`

`acronym("--computer Science--")` returns `"CS"`

`acronym("  university  texas computer  science---")` returns `"UTCS"`

`acronym("science   techNOLOGy Engineering      MATH")` returns `"STEM"`

`acronym("      cards AfFecTIng--BOARD       strategy")` returns `"CABS"`

`acronym(" - Ultimate BIG-  -- -ENDGAME  ramp  --     ")` returns `"UBER"`

In your answer you may use `String` concatenation and the `String` `length`, `charAt`, and `toUpperCase` methods.

You may not use any other methods or Java classes.

<u>This means you may NOT use the Scanner class.</u>

```
// Assume phrase contains at least one 'word' consisting of English
// letters and that the only characters in phrase are English
// letters, spaces, and dashes.
public static String acronym(String phrase) {
```

7. 2d Arrays - 15 points. Write a method `indexOfMaxPointsDifference` that returns the index of the team with the maximum points difference for women's basketball teams and scores.

The score for various women's basketball teams are stored in a 2d array of ints.

The 2d array is square with the number of columns equal to the number of rows.

Each team's results are stored in a row and corresponding column.

So for example UT uses row 0 and column 0 in the example below.

The values in the team's row indicate the points scored by the team and the values in the team's column indicate the points scored by opposing teams.

Consider this small example with 5 teams and 8 games.

UT: 80 vs. A&M: 60
UT: 100 vs. TCU: 60
UT: 100 vs. SMU: 70
SMU: 50 vs. A&M: 60
TCU: 50 vs. SMU: 80 .
UT: 60 vs. UNT: 50
A&M: 100 vs. UNT: 50
TCU: 60 vs. A&M: 80

		UT	A&M	TCU	SMU	UNT
		0	1	2	3	4
UT	0	-1	80	**100**	100	60
A&M	1	60	-1	80	60	100
TCU	2	**60**	60	-1	50	-1
SMU	3	70	50	80	-1	-1
UNT	4	50	50	-1	-1	-1

Consider the game between UT and TCU.

Assume UT is row 0 and TCU is row 2 in the score matrix.

UT's score of 100 is stored in cell (0, 2) and TCU's score of 60 is stored in cell (2, 0).

The matrix above shows the row and column numbers, 0 through 4. The team names are shown for illustration purpose only. They will not actually be present in the data sent to the method.

A team's point difference is the sum of the points scored by the team minus the points scored against the team.

So for example UT's point difference is (80 + 100 + 100 + 60) - (70 + 60 + 70 + 50) = 340 - 250 = 90

A `-1` in a cell indicates a game did not take place between the teams.

Your method shall return the index of the team that has the maximum point difference. If there is a tie for team's with the maximum point difference return the index closest to 0.

You may not use any other Java classes or methods other than the array length field.

Do not create any new arrays.

```
/* assume scores.length >= 2 and that scores is a square matrix
     as described in the question. */
public static int indexOfMaxPointsDifference(int[][] scores) {
```

8. ArrayLists - 10 points. Write a method that given an `ArrayList` that contains `Double` objects, removes all values from the `ArrayList` that are greater than or equal to twice the average of the values in the initial `ArrayList`.

For example if we had the following `ArrayList`:

```
[10.0, 20.0, 10.0, 100.8, 70.0, 40.0, 30.0, -10.0]
```

The average of the values in the `ArrayList` is 270.8 / 8 = 33.85. Twice the average is 67.7.

The values of 100.8 and 70.0 shall be removed from the `ArrayList` resulting in

```
[10.0, 20.0, 10.0, 40.0, 30.0, -10.0]
```

The relative order of the other items must remain the same.

Recall the following methods from the `ArrayList` class.

These are the only method you may use in your answer.

```
int size() Number of elements in the list.

get(int pos)  Returns the element at the given position.

remove(int pos) Remove and return the value at the given position.
```

Recall the `Double` class is a "wrapper" class. It stores a single `double` value. Java will auto wrap and unwrap doubles. So given an `ArrayList<Double> list` the following code compiles and works as intended.

```
double a = list.get(2); // assuming size >= 2
double b = 10.75;
list.add(b / 2);
```

Do not create or use any other data structures such as arrays or other `ArrayLists`.

```
public static void removeValuesTwiceAverage(ArrayList<Double> list) {
```

9. File Procession - 20 Points. Write the method `printAPScoreAverage`. The method accepts three parameters, a `Scanner` already connected to a data source, an `int` representing a grade level, and a `boolean` that indicates whether to analyze STEM (Science, Technology, Engineering, Math) scores or non-STEM scores.

The Scanner sent to the method is connected to a data source representing high school students' Advanced Placement (AP) test scores.

```
10 Mike Scott

12 Sonika Garg
5 5 * 5 * 5 5 4 * 5 * 4
12 Shelby McGarrah
5 * 5 * 5 * 5 *
10 C K Khatri
1 1 5 4
```

Each set of data consists of two lines in the file.

`12 Sonika Garg`	Grade level followed by name
`5 5 * 5 * 5 5 4 5 *`	AP scores

The first line of a data set contains the student's grade level represented as an integer. The rest of the first line is the student's name.

The second line lists the AP scores for the student. The scores will range from 1 to 5. Scores followed by an asterisk, *, indicate the test was a STEM area such as Computer Science or Statistics. Scores that are not followed by an asterisk indicate the test was a non-STEM area such as French or Studio Art.

Complete a method that prints out the average AP score for students in the given grade and for the given area (STEM or non-STEM). Consider the following examples given the file above:

`printAPScoreAverage(sc, 11, true)` results in no output because there are no students in grade 11 in the example data file.

`printAPScoreAverage(sc, 10, true)` results in the following output:

```
Mike Scott: NO EXAMS
C K Chatri: NO EXAMS
```

`printAPScoreAverage(sc, 10, false)` results in the following output:

```
Mike Scott: NO EXAMS
C K Chatri: 2.75
```

`printAPScoreAverage(sc, 12, true)` results in the following output:

```
Sonika Garg: 4.75
12 Shelby McGarrah: 5.0
```

Complete the following method. You may use the `hasNext, hasNextLine, hasNextInt, next, nextLine,` and `nextInt` methods from the `Scanner` class.
You may create and use new `Scanner` objects. Do not use any other Java classes or methods.

HINT: Create and use a helper method that given a String with the line of test data for the student calculates and returns the average score for STEM or non STEM exams. Use the next page for the helper method.

```
public static void printAPScoreAverage(Scanner sc, int grade,
                                                    boolean STEM) {
```

```
// Room for helper method.
```