1. Expressions (10 points)

For each expression in the left-hand column, indicate its value in the right-hand column. Be sure to list a constant of appropriate type (e.g., `7.0` rather than `7` for a `double`, `String`s in quotes, `true`/`false` for a `boolean`). You do not need to show work.

Expression	Value
`1 + 2 * 3 - 4 / (5 - 1)`	_____
`14 + 40 % 7 % 4 * 3 - 20 % (4 + 2)`	_____
`25 / (5 / 2.0) - 14 * 0.5 / 2`	_____
`1 + " + 2 + " + 2 + 1 + "1" + 2`	_____
`24 / 2 / 5 / 2.0 + 0.1 * 2 + 3 / 2 / 1`	_____

Answers:

`1 + 2 * 3 - 4 / (5 - 1) =`	`6`
`14 + 40 % 7 % 4 * 3 - 20 % (4 + 2) =`	`15`
`25 / (5 / 2.0) - 14 * 0.5 / 2 =`	`6.5`
`1 + " + 2 + " + 2 + 1 + "1" + 2 =`	`"1 + 2 + 2112"`
`24 / 2 / 5 / 2.0 + 0.1 * 2 + 3 / 2 / 1 =`	`2.2`

2. Parameter Mystery (12 points)

At the bottom of the page, write the output produced by the following program.

```
public class ParameterMystery {
    public static void main(String[] args) {
        String jake = "jake";
        String finn = "hero";
        String princess = "bubblegum";
        String simon = "iceking";

        beemo(jake, finn, princess);
        beemo(simon, princess, simon);
        beemo(jake, "finn", finn);
        beemo(simon, "bubblegum", "jake");
    }

    public static void beemo(String finn, String hero, String princess) {
        System.out.println(hero + " and " + princess + " are totally " + finn);
    }
}
```

Answers:

```
hero and bubblegum are totally jake
bubblegum and iceking are totally iceking
finn and hero are totally jake
bubblegum and jake are totally iceking
```

3. If/Else Simulation (12 points)

Given the method definition, for each method call in the left-hand column, write the output that is produced by the program in the right-hand column. You do not need to show work.

```
public static void mystery(int a, int b) {
    int c = 2;
    if (a + c < b) {
        c = c + 8;
    } else {
        b = b + 10;
    }
    if (a + c < b) {
        c = c + 8;
    } else {
        b = b + 10;
    }
    System.out.println(b + " " + c);
}
```

Method call	Value
`mystery(5, 100)`	_____
`mystery(2, 6)`	_____
`mystery(20, 4)`	_____
`mystery(10, 12)`	_____

Answers:
100 18
16 10
24 2
22 10

4. While Loop Simulation (12 points)

Given the method definition, for each method call in the left-hand column, write the value returned in the right-hand column. You do not need to show work.

```
public static int mystery(int z) {
    int x = 1;
    int y = 1;
    while (z > 2) {
        y = y + x;
        x = y - x;
        z--;
    }
    return y;
}
```

Method call	Value
`mystery(3)`	_____
`mystery(4)`	_____
`mystery(-1)`	_____
`mystery(6)`	_____

Answers:
2
3
1
8

5. Assertions (15 points)

For the following method, identify each of the three assertions in the table below as being one of ALWAYS true, NEVER true or SOMETIMES true / sometimes false at each labeled point in the code. (You may abbreviate these choices as A/N/S respectively.)

```
public static int mystery(int max) {
    Random rand = new Random();
    int height = 0;
    int falls = 0;

    // Point A
    while (height < max) {
        int r = rand.nextInt(4);

        // Point B
        if (r == 0 && height > 0) {
            height--;
            falls++;
            // Point C
        } else {
            height++;
            // Point D
        }
    }

    // Point E
    return falls;
}
```

	falls == 0	height > 0	height < max
Point A	A	N	S
Point B	S	S	A
Point C	N	S	A
Point D	S	A	S
Point E	S	S	N

6. Programming (15 points)

Write a method called `splitTheBill` that interacts with a user to split the total cost of a meal evenly. First ask the user how many people attended, then ask for how much each of their meals cost. Finally, print out a message to the user indicating how much everyone has to pay if they split the bill evenly between them.

Round the split cost to the nearest cent, even if this means the restaurant gets a couple more or fewer cents than they are owed. Assume that the user enters valid input: a positive integer for the number of people, and real numbers for the cost of the meals.

Sample logs, user input **bold and underlined**:

```
How many people? 4
Cost for person 1: 12.03
Cost for person 2: 9.57
Cost for person 3: 17.82
Cost for person 4: 11.07

The bill split 4 ways is: $12.62
```

```
How many people? 1
Cost for person 1: 87.02

The bill split 1 ways is: $87.02
```

```
How many people? 6
Cost for person 1: 5.03
Cost for person 2: 4.86
Cost for person 3: 7.23
Cost for person 4: 4.07
Cost for person 5: 6.84
Cost for person 6: 4.95

The bill split 6 ways is: $5.50
```

Answer:

```
    public static void splitTheBill(Scanner console) {
        System.out.print("How many people? ");
        int people = console.nextInt();

        double total = 0;
        for (int i = 1; i <= people; i++) {
            System.out.print("Cost for person " + i + ": ");
            total += console.nextDouble();
        }
        double average = total / people;
        average = Math.round(average * 100) / 100.0;

        System.out.println();
        System.out.println("The bill split " + people + " ways is : $" + average);
    }
```

7. Programming (15 points)

Write a method called `walkHome` that traces the path of a star-shaped critter (**`*`**) on its way back to its home (**`|^|`**), and returns how many steps it took to arrive. The critter starts out 4 units away from its house, and repeatedly takes steps towards or away from its destination. It knows where home is, but it is a little lost. It's movement is random, but on average it moves towards home 3/5 of the time, and moves away from home 2/5 of the time.

Your method should produce a line of output for the starting position, and then for every step the critter takes until it reaches home. These repeated lines should show the house, the critter, and dashes (**`-`**) representing the empty spaces in between the critter and its house. Once the critter reaches home, it should display the house and the critter right next to each other, with no dashes in between (**`|^|*`**).

Your method takes a Random object as a parameter, and you should use said Random object to decide whether the critter steps closer to or farther from home. Since there is randomness, your output will likely not match these samples exactly, but the format should match.

Sample call:

```
Random r = new Random();
int steps = walkHome(r);   //returns 5
```

Output:

```
|^|-----*
|^|----*
|^|---*
|^|--*
|^|-*
|^|*
```

Sample call:

```
Random r = new Random();
int steps = walkHome(r); // returns 11
```

Output:

```
|^|-----*
|^|------*
|^|-----*
|^|------*
|^|-----*
|^|----*
|^|---*
|^|--*
|^|-*
|^|--*
|^|-*
|^|*
```

Answer:

```
    public static int walkHome(Random r) {
        int position = 5;

        int steps = 0;
        while (position > 0) {
            steps++;

            System.out.print("|^|");
            for (int i = 0; i < position; i++) {
                System.out.print("-");
            }
            System.out.println("*");

            int random = r.nextInt(5);
            if (random == 0 || random == 1) {
                position++;
            } else {
                position--;
            }
        }
        System.out.println("|^|*");
        return steps;
    }
```

8. Programming (9 points)

Write a method called `numWords` that takes a `String` as a parameter and that returns the number of words in the `String`. By definition, words are separated by one or more spaces. The table below shows several sample calls and the value that should be returned.

Example calls:	Value returned:
`numWords("how  many  words  here?")`	4
`numWords("to be or not to be, that is the question")`	10
`numWords("  how  about  merry-go-round  ")`	3
`numWords(" !&$%--$$!!*()  foo_bar_baz  ")`	2
`numWords("x")`	1
`numWords("  ")`	0
`numWords("")`	0

Notice that words can contain punctuation marks. Any non-empty sequence of non-space characters can be a word. Also notice that there might be spaces at the beginning or end of the `String`.

You may not construct any other objects to solve this problem (e.g., you can't use a `Scanner` or tokenizer). You may assume that the `String` has no other whitespace characters such as tabs or newline characters. Your method can pay attention just to spaces to decide how many words there are.

One possible answer:

```
    public static int numWords(String s) {
        int count = 0;
        boolean inWord = false;

        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == ' ')) {
                inWord = false;
            } else if (!inWord) {
                count++;
                inWord = true;
            }
        }
        return count;
    }
```