CSE 142 SPRING 2011 MIDTERM SOLUTIONS

1. Expressions (9 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"). If the expression is illegal, then write "error".

Expression	Value
`1 + 2 * (4 + 3) - 5 * 2`	`5`
`11 % 6 + 20 % 3 % 2 + 1 % 2`	`6`
`3 * 2 + 1 + 13 / 3 / 2 * 2`	`11`
`0.5 + 2 / 4 - 0.25 * 2 + "7"`	`"0.07"`

2. Parameter Mystery (15 points)

At the bottom of the page, write the output produced by the following program, as it would appear on the console.

```
public class ParameterMystery {
    public static void main(String[] args) {
        String what = "pinky";
        String want = "brain";
        String tonight = "narf";
        String you = "tonight";
        String pinky = "egad";
        String brain = "what";
        String narf = "want";

        mystery(what, "want", "tonight");
        mystery("egad", want, tonight);
        mystery(you, brain, narf);
        mystery(pinky, "and the", brain);
    }

    public static void mystery(String tonight, String what, String want) {
        System.out.println(what + " do you " + want + " to do " + tonight);
    }
}
```

```
want do you tonight to do pinky
brain do you narf to do egad
what do you want to do tonight
and the do you what to do egad
```

3. While Loop Simulation (15 points)

For each call below to the following method, write the output that is printed, as it would appear on the console:

```
public static void mystery(int x, int y, int z) {
    while (x >= y && x > z) {
        x = x - z;
        y--;
        z++;
    }
    System.out.println(x + " " + y + " " + z);
}
```

Method Call	Output
`mystery(31, 4, 15);`	`16 3 16`
`mystery(8, 4, 2);`	`3 2 4`
`mystery(0, 0, 7);`	`0 0 7`
`mystery(9, 1, 1);`	`3 -2 4`

4. Conditionals (9 points)

Consider the following piece of code where `x` is an integer:

```
if (x >= 2) {
    System.out.println("Pinky");
} else if (x <= 4) {
    System.out.println("Brain");
} else if (x >= 2 && x <= 4) {
    System.out.println("Narf!");
}
```

(a) Under what conditions does the code print out `"Pinky"`? Explain your answer in 50 words or less.

When `x` is greater than or equal to 2, because the first test will be true and the first branch will execute.

(b) Under what conditions does the code print out `"Narf!"`? Explain your answer in 50 words or less.

Never. To enter the third branch, one of the conditions is that `x` be greater than or equal to 2. If that were true, the first test would be true and the first branch would be executed instead.

(c) What is the largest value of `x` in which the code will print out `"Brain"`?

1. If `x` were 2, 3, or 4, the first branch would be executed instead.

5. Assertions (15 points)

For each of the five points labeled by comments, identify each of the following assertions as being either always true, never true or sometimes true / sometimes false.

```
public static int mystery(int x) {
    Scanner console = new Scanner(System.in);
    System.out.print("Enter a number GREATER than " + x + ": ");
    int y = console.nextInt();
    y = y * 2;

    // Point A
    while (y > x) {
        // Point B
        if (x % 2 == 1) {
            // Point C
            x++;
            y--;
        } else if (y % 2 == 0) {
            y /= 2;
        } else {
            y++;
            x = x - 2;
            // Point D
        }
    }

    // Point E
    return x;
}
```

Fill in each box of the the table below with one of the following words: ALWAYS, NEVER or SOMETIMES. (You may abbreviate these choices as A, N, and S as long as you write legibly.)

	`x % 2 == 0`	`y > x`	`y % 2 == 0`
Point A	SOMETIMES	SOMETIMES	ALWAYS
Point B	SOMETIMES	ALWAYS	SOMETIMES
Point C	NEVER	ALWAYS	ALWAYS*
Point D	ALWAYS	ALWAYS	ALWAYS
Point E	SOMETIMES	NEVER	SOMETIMES

***** This is a very subtle point. In fact, during grading, we had "SOMETIMES" as the official answer.

6. Programming (18 points)

Write a static method named `guess` that accepts two integers as parameters: a number to guess and how many tries the method gets to guess the number. The method will guess random numbers in the range of 1 to 10 (inclusive) and print them as it guesses them. If the method guesses the number, it will print `"I got it!"` and then return `true`. If it fails to guess the number in the requisite number of tries, it will print `"I give up!"` and then return `false`. You may assume that the number of tries given is a non-negative number.

Call	Prints	Returns
`guess(2, 5);`	`7 10 4 7 6 I give up!`	`false`
`guess(5, 0);`	`I give up!`	`false`
`guess(1, 10);`	`2 1 I got it!`	`true`
`guess(-3, 3);`	`9 6 2 I give up!`	`false`
`guess(6, 6);`	`10 10 3 6 I got it!`	`true`

As this method has an element of randomness to it, you are to copy the format of the output, not the exact output of the sample calls.

```
public static boolean guess(int answer, int tries) {
   Random rand = new Random();

   for (int i = 1; i <= tries; i++) {
      int num = rand.nextInt(10) + 1;
      System.out.print(num + " ");

      if (num == answer) {
         System.out.println("I got it!");
         return true;
      }
   }

   System.out.println("I give up!");
   return false;
}
```

7. Programming (18 points)

(a) (8 points) Write a static method named `nextWord` that takes a String parameter (one of `"rock"`, `"paper"` or `"scissors"`) and returns the next word in the sequence. If the word is `"scissors"`, then `nextWord` returns `"rock"`. The table below shows all possible calls to `nextWord`. You may assume that no other String will be passed to `nextWord`.

Call	Returns
`nextWord("rock");`	`"paper"`
`nextWord("paper");`	`"scissors"`
`nextWord("scissors");`	`"rock"`

```
public static String nextWord(String word) {
   if (word.equals("rock")) {
      return "paper";
   } else if (word.equals("paper")) {
      return "scissors";
   } else {
      return "rock";
   }
}
```

(b) (10 points) Using `nextWord`, write a static method named `rockPaperScissors` that takes a number as parameter. The given number represents the maximum number of Strings to print. The method will alternate printing out the Strings `"rock"`, `"paper"` and `"scissors"` on successive lines with the first line having one word, the second line having two words, the third line having three words, and so on. The total number of words printed should be no more than the given parameter. **No partial lines will be printed**, i.e. if printing out the line will cause the number of words to exceed the given number, then the line will not be printed. The sample calls below should make this clear.

`rockPaperScissors(5)`	`rockPaperScissors(6)`	`rockPaperScissors(17)`

`rock`
`paper scissors`
	`rock`
	`paper scissors`
	`rock paper scissors`
		`rock`
		`paper scissors`
		`rock paper scissors`
		`rock paper scissors rock`
		`paper scissors rock paper scissors`

Notice that `rockPaperScissors(5)` does not print out the third line. Doing so would require printing 3 more words, which would mean 6 words would have been printed--that would exceed the given number `5`. Passing `6` prints three lines (as would passing `7`, `8`, or `9`). Passing `10` would add the fourth line.

```
public static void rockPaperScissors(int count) {
   int currentMax = 1;

   String word = "rock";
   while (count >= currentMax) {
      for (int j = 1; j <= currentMax; j++) {
         System.out.print(word + " ");
         word = nextWord(word);
      }
      System.out.println();

      count -= currentMax;
      currentMax++;
   }
}
```