Sample Midterm Exam #1 Key

Also check out Practice-It to test solving these problems or to type in your own solution to see if it works!

1. Expressions

Expression	Value
`3 * 4 + 5 * 6 + 7 * -2`	`28`
`1.5 * 2.0 + (5.5 / 2) + 5 / 4`	`6.75`
`23 % 5 + 31 / 4 % 3 - 17 % (16 % 10)`	`-1`
`"1" + 2 + 3 + "4" + 5 * 6 + "7" + (8 + 9)`	`"123430717"`
`345 / 10 / 3 * 55 / 5 / 6 + 10 / (5 / 2.0)`	`24.0`
`1 / 2 > 0 || 4 == 9 % 5 || 1 + 1 < 1 - 1`	`true`

2. Parameter Mystery

```
tyler and tv like java
java and tyler like tv
tv and donnie like rugby
hamburger and x like tyler
tyler and java like tyler
```

3. If/Else Simulation

Method Call	Value Returned
`mystery(4, 2)`	`4`
`mystery(5, 4)`	`5`
`mystery(5, 13)`	`15`
`mystery(5, 17)`	`20`
`mystery(4, 8)`	`8`

4. While Loop Simulation

Method Call	Output
`mystery(5, 0);`	`5`
`mystery(3, 2);`	`1 0 1`
`mystery(16, 5);`	`3 2 1 0 1`
`mystery(80, 9);`	`8 4 2 1 2 0 2`
`mystery(1600, 40);`	`40 19 2 9 0 4 0`

5. Assertions

	y > x	z < 0	z > 0
Point A	SOMETIMES	NEVER	NEVER
Point B	NEVER	NEVER	SOMETIMES
Point C	SOMETIMES	NEVER	ALWAYS
Point D	ALWAYS	NEVER	SOMETIMES
Point E	ALWAYS	SOMETIMES	SOMETIMES

6. **Programming (five solutions shown)**

```
public static boolean hasMidpoint(int a, int b, int c) {
    double mid = (a + b + c) / 3.0;
    if (a == mid || b == mid || c == mid) {
        return true;
    } else {
        return false;
    }
}

public static boolean hasMidpoint(int a, int b, int c) {
    double mid = (a + b + c) / 3.0;
    return (a == mid || b == mid || c == mid);
}

public static boolean hasMidpoint(int a, int b, int c) {
    return (a == (b + c) / 2.0 || b == (a + c) / 2.0 || c == (a + b) / 2.0);
}

public static boolean hasMidpoint(int a, int b, int c) {
    int max = Math.max(a, Math.max(b, c));
    int min = Math.min(a, Math.min(b, c));
    double mid = (max + min) / 2.0;

    return (a == mid || b == mid || c == mid);
}

public static boolean hasMidpoint(int a, int b, int c) {
    return (a - b == b - c || b - a == a - c || a - c == c - b);
}
```

7. **Programming (one solution shown)**

```
public static void sequenceSum(double limit) {
    if (limit >= 1) {
        System.out.print("1");
        int denomenator = 1;
        double sum = 1.0;
        while (sum < limit) {
            denomenator++;
            sum += 1.0 / denomenator;
            System.out.print(" + 1/" + denomenator);
        }
        System.out.printf(" = %.3f\n", sum);
    }
}
```

8. Programming (three solutions shown)

```
public static void favoriteLetter(Scanner console, String letter) {
    System.out.println("Looking for two \"" + letter + "\" words in a row.");
    int count = 0;
    String word = "";
    while (count < 2) {
        System.out.print("Type a word: ");
        word = console.next();
        if (word.startsWith(letter)) {
            count++;
        } else {
            count = 0;
        }
    }
    System.out.println("\"" + letter + "\" is for \"" + word + "\"");
}

// uses two Strings instead of count, and uses forever/break loop
public static void favoriteLetter(Scanner console, String letter) {
    System.out.println("Looking for two \"" + letter + "\" words in a row.");
    System.out.print("Type a word: ");
    String word1 = console.next();
    System.out.print("Type a word: ");
    String word2 = console.next();
    while (!(word1.startsWith(letter) && word2.startsWith(letter))) {
        word1 = word2;
        System.out.print("Type a word: ");
        word2 = console.next();
    }
    System.out.println("\"" + letter + "\" is for \"" + word2 + "\"");
}

// uses do/while loop
public static void favoriteLetter(Scanner console, String letter) {
    System.out.println("Looking for two \"" + letter + "\" words in a row.");
    int count = 0;
    String word;
    do {
        System.out.print("Type a word: ");
        word = console.next();
        if (word.startsWith(letter)) {
            count++;
        } else {
            count = 0;
        }
    } while (count < 2);
    System.out.println("\"" + letter + "\" is for \"" + word + "\"");
}
```

Sample Midterm Exam #1

1. **Expressions**
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`).

Expression	Value
`3 * 4 + 5 * 6 + 7 * -2`	____________________________
`1.5 * 2.0 + (5.5 / 2) + 5 / 4`	____________________________
`23 % 5 + 31 / 4 % 3 - 17 % (16 % 10)`	____________________________
`"1" + 2 + 3 + "4" + 5 * 6 + "7" + (8 + 9)`	____________________________
`345 / 10 / 3 * 55 / 5 / 6 + 10 / (5 / 2.0)`	____________________________
`1 / 2 > 0 || 4 == 9 % 5 || 1 + 1 < 1 - 1`	____________________________

2. Parameter Mystery

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

```
public class ParameterMystery {
    public static void main(String[] args) {
        String x = "java";
        String y = "tyler";
        String z = "tv";
        String rugby = "hamburger";
        String java = "donnie";

        hamburger(x, y, z);
        hamburger(z, x, y);
        hamburger("rugby", z, java);
        hamburger(y, rugby, "x");
        hamburger(y, y, "java");
    }

    public static void hamburger(String y, String z, String x) {
        System.out.println(z + " and " + x + " like " + y);
    }
}
```

3. **If/Else Simulation**
For each call of the method below, write the value that is returned:

```
public static int mystery(int a, int b) {
    int c;
    if (a > b) {
        c = a;
    } else if (b % a == 0) {
        c = b;
    } else {
        c = b + (a - (b % a));
    }
    return c;
}
```

Method Call	Value Returned
mystery(4, 2)	________________________________
mystery(5, 4)	________________________________
mystery(5, 13)	________________________________
mystery(5, 17)	________________________________
mystery(4, 8)	________________________________

4. While Loop Simulation

For each call of the method below, write the output that is printed:

```
public static void mystery(int i, int j) {
    while (i != 0 && j != 0) {
        i = i / j;
        j = (j - 1) / 2;
        System.out.print(i + " " + j + " ");
    }
    System.out.println(i);
}
```

Method Call	Output
`mystery(5, 0);`	________________________________
`mystery(3, 2);`	________________________________
`mystery(16, 5);`	________________________________
`mystery(80, 9);`	________________________________
`mystery(1600, 40);`	________________________________

5. **Assertions**
For the following method, identify each of the three assertions in the table below as being either 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 x) {
    int y = 1;
    int z = 0;

    // Point A
    while (y <= x) {
        // Point B
        y = y * 10;
        z++;

        // Point C
    }

    // Point D
    z--;

    // Point E
    return z;
}
```

	y > x	z < 0	z > 0
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming

Write a static method named `hasMidpoint` that accepts three integers as parameters and returns `true` if one of the integers is the midpoint between the other two integers; that is, if one integer is exactly halfway between them. Your method should return `false` if no such midpoint relationship exists.

The integers could be passed in any order; the midpoint could be the 1st, 2nd, or 3rd. You must check all cases.

Calls such as the following should return `true` :

```
hasMidpoint(4, 6, 8)
hasMidpoint(2, 10, 6)
hasMidpoint(8, 8, 8)
hasMidpoint(25, 10, -5)
```

Calls such as the following should return `false` :

```
hasMidpoint(3, 1, 3)
hasMidpoint(1, 3, 1)
hasMidpoint(21, 9, 58)
hasMidpoint(2, 8, 16)
```

7. Programming

Write a static method named `sequenceSum` that prints terms of the following mathematical sequence:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\ldots \qquad \text{(also written as } \sum_{i=1}^{\infty}\frac{1}{i} \text{)}$$

Your method should accept a real number as a parameter representing a limit, and should add and print terms of the sequence until the sum of terms meets or exceeds that limit. For example, if your method is passed 2.0, print terms until the sum of those terms is at ≥ 2.0. You should round your answer to 3 digits past the decimal point.

The following is the output from the call `sequenceSum(2.0);`

```
1 + 1/2 + 1/3 + 1/4 = 2.083
```

(Despite the fact that the terms keep getting smaller, the sequence can actually produce an arbitrarily large sum if enough terms are added.) If your method is passed a value less than 1.0, no output should be produced. You must match the output format shown exactly; note the spaces and pluses separating neighboring terms. Other sample calls:

Calls	`sequenceSum(0.0);`	`sequenceSum(1.0);`	`sequenceSum(1.5);`
Output		`1 = 1.000`	`1 + 1/2 = 1.500`

Call	`sequenceSum(2.7);`
Output	`1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 = 2.718`

8. Programming

Write a static method named `favoriteLetter` that accepts two parameters: a `Scanner` for the console, and a favorite letter represented as a one-letter `String`. The method repeatedly prompts the user until two consecutive words are entered that start with that letter. The method then prints a message showing the last word typed.

You may assume that the user will type a single-word response to each prompt. Your code should be case-sensitive; for example, if the favorite letter is **a**, you should not stop prompting if the user types words that start with an **A**. For example, the following logs represent the output from two calls to your method: (User input is underlined.)

Call

```
Scanner console = new Scanner(System.in);
favoriteLetter(console, "y");
```

```
Scanner console = new Scanner(System.in);
favoriteLetter(console, "A");
```

Output

```
Looking for two "y" words in a row.
Type a word: hi
Type a word: bye
Type a word: yes
Type a word: what?
Type a word: yellow
Type a word: yippee
"y" is for "yippee"
```

```
Looking for two "A" words in a row.
Type a word: I
Type a word: love
Type a word: CSE142!
Type a word: AND
Type a word: PROGRAMS
Type a word: are
Type a word: always
Type a word: Absolutely
Type a word: Awesome
"A" is for "Awesome"
```