Solution to CSE143 Section #10 Problems

1. One possible solution appears below.

```
public int factorial(int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    }

    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
```

2. One possible solution appears below.

```
public void parenthesize(String str, int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    }

    if (n == 0) {
        System.out.print(str);
    } else {  // n > 0
        System.out.print("(");
        parenthesize(str, n - 1);
        System.out.print(")");
    }
}
```

3. Two possible solutions appear below. The second is more efficient because it performs only one recursive call to compute the string to be concatenated with itself.

```
public String starString(int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    }

    if (n == 0) {
        return "*";
    } else {
        return starString(n - 1) + starString(n - 1);
    }
}

public String starString(int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
```

```
        }

        if (n == 0) {
            return "*";
        } else {
            String half = starString(n - 1);
            return half + half;
        }
    }
```

4. One possible solution appears below.

```
      public void writeNums(int n) {
          if (n < 1) {
              throw new IllegalArgumentException();
          }

          if (n == 1) {
              System.out.print(1);
          } else {
              writeNums(n - 1);
              System.out.print(", " + n);
          }
      }
```

5. One possible solution appears below.

```
    public void writeChars(int n) {
        if (n < 1) {
            throw new IllegalArgumentException();
        }

        if (n == 1) {
            System.out.print("*");
        } else if (n == 2) {
            System.out.print("**");
        } else {
            System.out.print("<");
            writeChars(n - 2);
            System.out.print(">");
        }
    }
```

6. One possible solution appears below.

```
    public void printTwos(int n) {
        if (n < 1) {
            throw new IllegalArgumentException();
        }

        if (n % 4 == 0) {
            System.out.print("2 * ");
            printTwos(n / 4);
```

```
            System.out.print(" * 2");
        } else if (n % 2 == 0) {
            System.out.print("2 * ");
            printTwos(n / 2);
        } else {
            System.out.print(n);
        }
    }
```

7. One possible solution appears below.

```
    public void stutter(Stack<Integer> s) {
        if (!s.isEmpty()) {
            int next = s.pop();
            stutter(s);
            s.push(next);
            s.push(next);
        }
    }
```

8. One possible solution appears below.

```
    public void writeSquares(int n) {
        if (n < 1) {
            throw new IllegalArgumentException();
        }

        if (n == 1) {
            System.out.print(1);
        } else if (n % 2 == 1) {
            System.out.print(n * n + ", ");
            writeSquares(n - 1);
        } else {
            writeSquares(n - 1);
            System.out.print(", " + n * n);
        }
    }
```

9. One possible solution appears below.

```
    public String substring(String s, int i, int j) {
        if (i < 0 || i > j || j > s.length()) {
            throw new IllegalArgumentException();
        }

        if (i == j) {
            return "";
        } else {
            return s.charAt(i) + substring(s, i + 1, j);
        }
    }
```

10. One possible solution appears below.

```
public void writeSequence(int n) {
    if (n < 1) {
        throw new IllegalArgumentException();
    }

    if (n == 1) {
        System.out.print(1);
    } else if (n == 2) {
        System.out.print("1 1");
    } else {
        int number = (n + 1) / 2;
        System.out.print(number + " ");
        writeSequence(n - 2);
        System.out.print(" " + number);
    }
}
```