CSE142 Final Exam Key
Autumn 2018

1. The program produces the following output:

```
2 [2, 4, 6, 0, 1] (0, 0)
3 [2, 4, 6, -35, 1]
2 [2, 4, 6, -35, 1]
1 (2, 1)
2 (2, 1)
```

2.

Original Array	Final Array
[10, 8, 6, 4, 2]	[10, 18, 24, 28, 30]
[1, 2, 3, 4, 5]	[1, 1, 2, 2, 3]
[5, 10, 8, 1, 3]	[5, 5, 3, 4, 7]
[4, 4, 4, 4, 4]	[4, 0, 4, 0, 4]
[19]	[19]

3. The program produces the following output:

```
Ralph 1
Felix 2   Ralph 2
Fix-It

Ralph 1
Ralph 2
Wreck-It

Vanellope 1   Vanellope 2
Vanellope 2
Glitch

Calhoun 2   Ralph 1    Calhoun 1
Calhoun 2   Ralph 1
Fix-It
```

4. One possible solution:

```
public static void processScores(Scanner input) {
    while (input.hasNextLine()) {
        String name = input.nextLine();
        String data = input.nextLine();
        int plus = 0;
        int count = 0;
        for (int i = 0; i < data.length(); i++) {
            count++;
            if (data.charAt(i) == '+') {
                plus++;
            }
        }
        double percent = 100.0 * plus / count;
        System.out.println(name + ": " + percent + "% plus");
    }
}
```

5. Two possible solutions:

```
public static double evaluate(Scanner input) {
    double result = input.nextDouble();
    while (input.hasNext()) {
        String operator = input.next();
        double num = input.nextDouble();
        if (operator.equals("+")) {
            result += num;
        } else {  // operator.equals("-")
            result -= num;
        }
    }
    return result;
}
```

```
public static double evaluate(Scanner input) {
    double sum = 0.0;
    boolean plus = true;
    while (input.hasNext()) {

        if (input.hasNextDouble()) {
                if (plus) {
                    sum += input.nextDouble();
            } else { // !plus (i.e. minus)
                        sum -= input.nextDouble();
                }
        } else {
            String sign = input.next();
            plus = sign.equals("+");

        }
    }
    return sum;
}
```

6. Four possible solutions:

```
public static boolean repeatedSequence(int[] a1, int[] a2) {
    if (a2.length % a1.length != 0) {
        return false;
    }

    for (int i = 0; i < a2.length; i++) {
        if (a2[i] != a1[i % a1.length]) {
            return false;
        }
    }
    return true;
}
```

```
public static boolean repeatedSequence(int[] a1, int[] a2) {
    if (a2.length % a1.length != 0) {
        return false;
    }

    for (int i = 0; i < a2.length; i += a1.length) {
        for (int j = 0; j < a1.length; j++) {
            if (a1[j] != a2[i + j]) {
                return false;
            }
        }
    }

    return true;
}
```

```
public static boolean repeatedSequence(int[] a1, int[] a2) {
    if (a2.length % a1.length != 0) {
        return false;
    }

    for (int i = 0; i < a2.length / a1.length; i++) {
        for (int j = 0; j < a1.length; j++) {
            if (a1[j] != a2[j + i * a1.length]) {
                return false;
            }
        }
    }
    return true;
}
```

```
public static boolean repeatedSequence(int[] a1, int[] a2) {
    if (a2.length % a1.length != 0) {
        return false;
    }

    for (int i = 0; i < a1.length; i++) {
        for (int j = i; j < a2.length; j += a1.length) {
            if (a1[i] != a2[j]) {
                return false;
            }
        }
    }
    return true;
}
```

7. Two possible solutions:

```
public boolean isSummer() {
    if (getMonth() < 6 || getMonth() > 9) {
        return false;
    } else if (getMonth() == 7 || getMonth() == 8) {
        return true;
    } else if (getMonth() == 6 && getDay() >= 21) {
        return true;
    } else if (getMonth() == 9 && getDay() <= 20) {
        return true;
    } else {
        return false;
    }
}
```

```
    public boolean isSummer() {
        double date = getMonth() + (getDay() / 31);
        return (date >= 6.677 && date <= 9.646);
    }
```

8. One possible solution:

```
    public class Rhino extends Critter {
        private int speed;
        private int moveCount;
        private Direction dir;
        private Random rand;

        public Rhino(Direction dir, int speed) {
            this.dir = dir;
            this.speed = speed;
            this.moveCount = 0;
            this.rand = new Random();
        }

        public Direction getMove() {
            if (moveCount >= speed) {
                moveCount = 0;
                int num = rand.nextInt(5);
                if (num == 0) {
                    dir = Direction.NORTH;
                } else if (num == 1) {
                    dir = Direction.EAST;
                } else if (num == 2) {
                    dir = Direction.SOUTH;
                } else if (num == 3) {
                    dir = Direction.WEST;
                } // dir stays the same if num == 5
            }

            moveCount++;
            return dir;
        }

        public boolean eat() {
            speed++;
            return true;
        }

        public Attack fight(String opponent) {
            speed--;
            if (speed < 1) {
                speed == 1
            }
            return Attack.POUNCE;
        }

        public Color getColor() {
            if (speed > 6) {
                return Color.WHITE;
            }
            return Color.GRAY;
        }
    }
```

9. Two possible solutions:

```
public static void banish(int[] a1, int[] a2) {
    for (int i = 0; i < a2.length; i++) {
        for (int j = 0; j < a1.length; j++) {
            if (a2[i] == a1[j]) {
                // remove element j from a1
                for (int k = j; k < a1.length - 1; k++) {
                    a1[k] = a1[k + 1];
                }
                a1[a1.length - 1] = 0;
                j--;
            }
        }
    }
}
```

```
public static void banish(int[] a1, int[] a2) {
    for (int i = 0; i < a1.length; i++) {
        // see whether a1[i] is contained in a2
        boolean found = false;
        for (int j = 0; j < a2.length && !found; j++) {
            if (a1[i] == a2[j]) {
                found = true;
            }
        }
        if (found) {
            // shift all elements of a1 left by 1
            for (int j = i + 1; j < a1.length; j++) {
                a1[j - 1] = a1[j];
            }
            a1[a1.length - 1] = 0;
            i--;   // so that we won't skip an index
        }
    }
}
```

10. Three possible solutions:

```
public static int collapseDigits(int value) {
    int result = 0;
    int count = 0;
    int prev = -1;

    while (value > 0) {
        int digit = value % 10;
        value = value / 10;

        if (digit != prev) {
            result += digit * Math.pow(10, count);
            count++;
            prev = digit;
        }
    }
    return result;
}
```

```
public static int collapseDigits(int n) {
    int result = 0;
    int place = 0;

    while (n >= 10) {
        int digit = n % 10;
        int rest = n / 10;
        if (digit == rest % 10) {
            n = rest / 10;
        } else {
            n = rest;
        }
        result += Math.pow(10, place) * digit;
        place++;
    }
    result += Math.pow(10, place) * n;
    return result;
}

public static int collapseDigits(int value) {
    int result = 0;
    int place = 0;

    while (value > 0) {
        result += value % 10 * Math.pow(10, place);

        if (value % 10 == value % 100 / 10) {
            value /= 100;
        } else {
            value /= 10;
        }
        place++;
    }
    return result;
}
```