CSE142 Final Exam Key
Summer 2018

1.

Original Array	Final Array
[10, 10, 10]	[11, 11, 10]
[2, 3, 4, 5]	[3, 5, 5, 5]
[3, 4, 5, 7, 9]	[3, 5, 7, 9, 10]
[2, 3, 5, 7, 9]	[3, 5, 7, 9, 10]
[4]	[4]

2. The program produces the following output:

```
soup
soup 1
stew 2    soup 1

soup
soup 1
soup 2

chowder
soup 2    chowder 1
soup 2

chili
chili 1
stew 2    chili 1
```

3. Three possible solutions:

```
public static void countCoins(Scanner input) {
      int totalCents = 0;
      while (input.hasNext()) {
            int count = input.nextInt();
            String coin = input.next().toLowerCase();
            if (coin.equals("nickels")) {
                  count *= 5;
            } else if (coin.equals("dimes")) {
                  count *= 10;
            } else if (coin.equals("quarters")) {
                  count *= 25;
            }
            totalCents += count;
      }

      int dollars = totalCents / 100;
      int cents = totalCents % 100;

      System.out.println("Total money: $" + dollars + "." cents);
}

public static void countCoins(Scanner input) {
      double total = 0.0;
      while (input.hasNext()) {
            int count = input.nextInt();
            String coin = input.next().toLowerCase();
            if (coin.equals("nickels")) {
                  count *= 5;
            } else if (coin.equals("dimes")) {
                  count *= 10;
            } else if (coin.equals("quarters")) {
                  count *= 25;
            }
            total += (double) count / 100;
      }
      System.out.println("Total money: $" + total);
}

public static void countCoins(Scanner input) {
      double total = 0.0;
      while (input.hasNext()) {
            double count = (double) input.nextInt();
            String coin = input.next().toLowerCase();

            if (coin.equals("pennies")) {
                  count *= 0.01;
            } else if (coin.equals("nickels")) {
                  count *= 0.05;
            } else if (coin.equals("dimes")) {
                  count *= 0.10;
            } else {
                  count *= 0.25;
            }
            total += count;
      }
      System.out.println("Total money: $" + total);
}
```

4. Three possible solutions:

```
public static int range(int[] a) {
      int min = a[0];
      int max = a[0];
      for (int i = 1; i < a.length; i++) {
            min = Math.min(min, a[i]);
            max = Math.max(max, a[i]);
      }
      return max - min;
}

public static int range(int[] a) {
      int min = 0;
      int max = 0;
      for (int i = 0; i < a.length; i++) {
            if (i == 0 || a[i] < min) {
                  min = a[i];
            }
            if (i == 0 || a[i] > max) {
                  max = a[i];
            }
      }

      int valueRange = max - min ;
      return valueRange;
}

public static int range(int[] a) {
      int range = 0;
      for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a.length; j++) {
                  int difference = Math.abs(a[i] - a[j]);
                  if (difference > range) {
                        range = difference;
                  }
            }
      }

      return range;
}
```

5. One possible solution:

```
public class Wombat extends Critter {
    private int count;
    private String display;
    private Random r;

    public Wombat() {
        display = "?";
        r = new Random();
    }

    public Direction getMove() {
        count++;
        if (count % 3 == 0) {
            return Direction.NORTH;
        } else {
            return Direction.SOUTH;
        }
    }

    public Attack fight(String opponent) {
        display = opponent;
        if (opponent.equals("%")) {
            return Attack.ROAR;
        }
        return Attack.POUNCE;
    }

    public boolean eat() {
        int prob = r.nextInt(3);
        return prob < 1;
    }

    public String toString() {
        return display;
    }
}
```

6. The program produces the following output:

```
1 7 [2, 4, 6]
0 8
1 7
-1 [2, 4, 6]
5 7 [2, 4, 6]
```

7. One possible solution:

```
public static void gradeQuiz(Scanner input) {
     int total = 0;
     int correct = 0;

     while (input.hasNextLine()) {
          total++;
          String question = input.nextLine();
          String correctAnswer = input.nextLine();
          String givenAnswer = input.nextLine();

          System.out.println("Question " + total + ": " + question);
          System.out.println("Student's response: " + givenAnswer);
          System.out.println("Correct answer: " + correctAnswer);

          if (givenAnswer.equalsIgnoreCase(correctAnswer)) {
               System.out.println("Correct!");
               correct++;
          } else {
               System.out.println("Incorrect. :-(");
          }
          System.out.println();
     }

     System.out.println(correct + "/" + total +
                        " questions answered correctly.");
}
```

8. Three possible solutions:

```
public boolean isLaterThan(ClockTime other) {
     if (this.getAmPm().equals(other.getAmPm())) {
          return (this.getHour() % 12 > other.getHour() % 12) ||
                     (this.getHour() == other.getHour() &&
                      this.getMinute() > other.getMinute());
     } else {
          return this.getAmPm().equals("PM");
     }
}

public boolean isLaterThan3(ClockTime other) {
     if (this.getAmPm().equals("PM") && other.getAmPm().equals("AM")) {
          return true;
     } else if (this.getAmPm().equals("AM") && other.getAmPm().equals("PM")) {
          return false;
     } else if (this.getHour() == other.getHour()) {
          return this.getMinute() > other.getMinute();
     } else if (this.getHour() == 12) {
          return false;
     } else if (other.getHour() == 12) {
          return true;
     } else {
          return this.getHour() > other.getHour();
     }
}

public boolean isLaterThan2(ClockTime other) {
     int totalMinutes1 = this.getHour() % 12 * 60 + this.getMinute();
     int totalMinutes2 = other.getHour() % 12 * 60 + other.getMinute();

     if (this.getAmPm().equals("PM")) {
          totalMinutes1 += (12 * 60);
     }
     if (other.getAmPm().equals("PM")) {
          totalMinutes2 += (12 * 60);
     }

     return totalMinutes1 > totalMinutes2;
}
```

9. Two possible solutions:

```
public static int[] compressTriplets(int[] arr) {
     int[] result = new int[(arr.length + 2) / 3];

     for (int i = 0; i < arr.length - 2; i += 3) {
          result[i / 3] = arr[i] + arr[i + 1] + arr[i + 2];
     }

     if (arr.length % 3 != 0) {
          result[result.length - 1] = arr[arr.length - 1];
     }
     if (arr.length % 3 == 2) {
          result[result.length - 1] += arr[arr.length - 2];
     }

     return result;
}

public static int[] compressTriplets(int[] arr) {
     // Creates an array of correct length (multiple of 3 case)
     int resultLength = arr.length / 3;

     // Creates an array of correct length (leftover case)
     if (arr.length % 3 != 0) {
          resultLength++;
     }

     // constructs a result array
     int[] result = new int[resultLength];

     // loops through parameter to fill result with tripet sums
     for (int i = 0; i < arr.length - 2; i += 3) {
          int first = arr[i];
          int second = arr[i + 1];
          int third = arr[i + 2];
          result[i / 3] = first + second + third;
     }

     // handles leftover case
     if (arr.length % 3 != 0) { // one or two left over
          result[result.length - 1] = arr[arr.length - 1];
          // two left over
          if (arr.length % 3 == 2) {
             result[result.length - 1] += arr[arr.length - 2];
          }
     }

     // returns result as array
     return result;
}
```

10. Four possible solutions:

```
public static int count(String target, String source) {
      target = target.toUpperCase();
      source = source.toUpperCase();
      int count = 0;
      for (int i = 0; i < source.length(); i++) {
            if (source.substring(i).startsWith(target)) {
                  count++;
            }
      }
      return count;
}

public static int count(String target, String source) {
      int count = 0;
      for (int i = 0; i < source.length() - target.length() + 1; i++) {
            String sub = source.substring(i, i + target.length());
            if (sub.equalsIgnoreCase(target)) {
                  count++;
            }
      }
      return count;
}

public static int count(String target, String source) {
      target = target.toUpperCase();
      source = source.toUpperCase();
      int count = 0;
      while (source.length() > 0) {
            int pos = source.indexOf(target);
            if (pos == -1) {
                  return count;
            } else {
                  count++;
                  source = source.substring(pos + 1);
            }
      }
      return count;
}

public static int count(String target, String source) {
      target = target.toUpperCase();
      source = source.toUpperCase();
      int count = 0;
      for (int i = 0; i < source.length() - target.length() + 1; i++) {
            boolean same = true;
            for (int j = 0; j < target.length(); j++) {
                  if (target.charAt(j) != source.charAt(i + j)) {
                        same = false;
                  }
            }
            if (same) {
                  count++;
            }
      }
      return count;
}
```