CS 139 Practice Midterm Questions #2

Spring 2016

Name: _____

1. Write Java statements to accomplish each of the following.
   (a) Declares `numbers` to be an array of `int`'s.

   (b) Initializes `numbers` to contain a reference to a new integer array of size 7.

   (c) Stores the value `3` in the last position of the array `numbers`. (Your answer should *not* use a literal value for the array index.)

2. Evaluate the following Java expressions or indicate the syntax errors they contain:
   (a) `true || true`

   (b) `true && False`

   (c) `true && (true || true)`

   (d) `false && (true || false)`

   (e) `NOT (true || (true or false))`

   (f) `(true || false) && (true || false)`

   (g) `(true || false) && (5 > 7)`

   (h) `(5 < 7) && (5 > 7)`

   (i) `5 < "7"`

   (j) `'A' < 'C'`

3. What would be printed by the following application (if it were compiled and executed properly)?

```
public class Shapes {
      public static void main(String[] args) {
          System.out.println("R1: " + isSquare(300, 10));
          System.out.println("R2: " + isSquare(20, 400));
          System.out.println("R3: " + isSquare(20, 20));
      }

      public static boolean isSquare(int width, int length) {
          return (width == length);
      }
}
```

4. What would be printed by the following application (if it were compiled and executed properly)? Is `isCircle` a good name for this method?

```
public class Shapes {
      public static void main(String[] args) {
          System.out.printf("C1: %s\n", isCircle(300, 10));
          System.out.printf("C2: %s\n", isCircle(20, 400));
          System.out.printf("C3: %s\n", isCircle(20, 20));
      }

      public static boolean isCircle(int width, int length) {
          boolean result;

          if (width == length) {
              result = true;
          } else {
              result = false;
          }

          return result;
      }
}
```

5. What would be printed by the following application (if it were compiled and executed properly)?

```
public class Worthless {
   public static void main(String[] args) {
      int age, answer, areacode, iq, zipcode;

      age = 67;
      areacode = 540;
      iq = 289;
      zipcode = 22801;

      answer = useless(age, areacode, iq, zipcode);
      System.out.println("ID: " + answer);
   }

   public static int useless(int a, int b, int c, int d) {
      int answer;

      answer = 0;

      if (d > 65) {
         answer = 1;
      }

      if (b == 540) {
         answer = 2;
      } else {
         if (c > 200) {
            answer = 5;
         }
      }

      return answer;
   }
}
```

6. For the question above, number the statements in the order in which they would be completed.

7. Consider the following fragment:

```
boolean canGraduate;

if ((credits >= 120) && (gpa >= 2.0)) {
    canGraduate = true;
} else {
    canGraduate = false;
}
```

Rewrite this fragment:

(a) Using nested if statements with a single relational operator in each.

(b) Without an else block.

(c) Reversing the "roles" of the if and else blocks (i.e., the if block should assign false to canGraduate).

(d) Without an if statement (i.e., using just assignment statements).

8. What would be printed by the following fragment (if it were compiled and executed properly)?

```
for (int k = 5; k > 0; k = k - 1) {
    System.out.printf("T minus %d\n", k);
}
```

9. Rewrite the for loop above as a while loop.

10. Rewrite it as a do-while loop.

11. What would be printed by the following fragment (if it were compiled and executed properly)?

```
int i;
int n;

n = 0;
for (i = 0; i <= 5; n++) {
   System.out.printf("%d", i);
}
```

12. What would be printed by the following fragment (if it were compiled and executed properly)?

```
int i;

i = 0;
while (i < 2) {
   System.out.print("+");
   i++;
}

while (i < 2) {
   System.out.print("-");
   i++;
}
```

13. What would be printed by the following fragment (if it were compiled and executed properly)?

```
for (int i = 0; i < 4; i++) {
   for (int j = 2; j >= i; j--) {
      System.out.printf("%d %d\n", i, j);
   }
}
```

14. Consider the code fragment below:

```
Die[] dice1;
Die[] dice2;

dice1 = new Die[4];
dice2 = dice1;

dice1[0] = new Die(1);

for (int i = 0; i < dice1.length; i++) {
   dice2[i] = dice1[0];
}

dice1[1] = new Die(3);

for (Die curDie : dice2) {
   System.out.println(curDie.getFace());
}
```

(a) How many array variables are declared in this code?

(b) How many arrays are created in this code? (Remember that arrays are created using the new keyword.)

(c) How many Die objects are created in this code?

(d) Draw a diagram illustrating the contents of memory after this fragment executes.

(e) What output will be produced by this code?

15. Complete the following method (using a loop):

```
/**
 * This method calculates the probability of tossing a coin n times and and
 * getting n heads
 *
 * The probability of getting a single head is 0.5. The probability of
 * getting two heads is 0.5*0.5. The probability of n heads is
 * 0.5*0.5*...*0.5 (where there are n terms in the product).
 *
 * @param n The number of tosses
 * @return The probability of n heads in n tosses
 */
public static double probabilityOfHeads(int n) {









}
```

16. Complete the following method (using a loop).

```
/**
 * This method finds the number of parentheses in a String
 *
 * @param text The String
 * @return The number of parentheses
 */
public static int numberOfParentheses(String text) {









}
```

17. Complete the following method. Use an enhanced for loop.

```
/**
 * This method returns true if the target string appears at least twice in
 * the provided array of strings. All comparisons are case-sensitive.
 *
 * @param strings - The array of Strings that must be searched
 * @param target - The String to search for.
 * @return true if the target appears at least twice, false otherwise
 */
public static boolean appearsTwice(String[] strings, String target) {









}
```

18. Complete the following method.

```
 /**
 * This method takes an integer array as an argument and returns a copy of
 * that array in which all negative values have been replaced by 0. THE
 * ORIGINAL ARRAY MUST NOT BE MODIFIED.
 *
 * @param data The array to copy
 * @return The new array (with no negative values)
 */
public static int[] nonNegative(int[] data) {









}
```

19. Complete the following method.

```
/**
 * This method takes an array of strings that may have some null elements. It
 * returns a new (possibly smaller) array with the null elements removed. For
 * example if the input array were {"A", null, "B"}, the return value would
 * be {"A", "B"}.
 *
 * THE ORIGINAL ARRAY MUST NOT BE MODIFIED.
 *
 * @param text The array of strings (possibly containing some null entries.)
 * @return A copy of the array without any null values.
 */
public static String[] removeNulls(String[] text) {
   // I suggest the following steps:
   //
   // 1. Use a loop to count the non-null entries. (This could be made into a
   // helper method.)
   // 2. Create a new array large enough to hold the non-null entries.
   // 3. Loop through the original array, copying the non-null entries into
   // the new array.









}
```