1. Short Answer (7 points)

(a) The = operator is used for assignment.
    The == operator is used for testing equality.

(b) boolean

(c) True

(d) 1.0

(e) 48A

(f) 10010001010

(g) 2212

2. Arrays, Conditionals, Loops, and Bugs (6 points)

Corrections are noted in on-line comments.

```java
public static void sort(int[] a) {
    int N = a.length;  // Add this line to declare and initialize N.
    for (int i = 0; i < N; i++) {  // Add this left curly brace.
        // find ith smallest
        int mini = i;  // Start mini at i each iteration.
        for (int j = i; j < N; j++)
            if (a[j] < a[mini])
                mini = j;

        // swap entries to put the ith smallest in the ith position
        // now a[0] ... a[i] have the correct entries
        int tmp = a[i];  // Add this line. tmp needed for swap.
        a[i] = a[mini];
    }
}
```
3. Recursion (6 points)

(a) The output of java Quest 11 is 4.
The calls are

\[
\begin{align*}
&BS(11, a, 0, 7) \\
&m = 3 \\
&BS(11, a, 4, 7) \\
&m = 5 \\
&BS(11, a, 4, 5) \\
&m = 4, \text{ since } a[4] \text{ is 11, return 4}
\end{align*}
\]

(b) The output of java Quest 12 is -1. The calls are

\[
\begin{align*}
&BS(12, a, 0, 7) \\
&m = 3 \\
&BS(12, a, 4, 7) \\
&m = 5 \\
&BS(12, a, 4, 5) \\
&m = 4 \\
&BS(12, a, 5, 5) \\
&m = 5, \text{ since } i==j \text{ return -1}
\end{align*}
\]

(c) If it finds the number x in the sorted array, BS returns the appropriate index, otherwise it returns -1. BS stands for Binary Search which is a process of splitting a sorted list in half and looking in the lower or upper half.

4. Combinational Logic (8 points)

(a) Truth table for the requested function f, as well as the decimal equivalent of X:

<table>
<thead>
<tr>
<th>abc</th>
<th>decimal</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>-3</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) \( f = a'bc + ab'c' + ab'c \)

(c) The circuit has three NOT gates, to provide \( a' \), \( b' \), and \( c' \).

Three 3-input AND gates have inputs matching the terms in the equation, and output going to a 3-input OR gate, whose output is f.

(d) The term \( ab'c' + ab'c \) is independent of \( c \), and thus equivalent to \( ab' \).

The simplified equation is \( f = a'bc + ab' \).

The simplified circuit has two NOT gates, two AND gates, one OR gate.
5. Java Programming and Graphics (9 points)

(a) Write the `drawFilledCircle` method.

```java
public static void drawFilledCircle(double x, double y, double radius, Color color) {
    StdDraw.penUp();
    StdDraw.go(x, y);
    StdDraw.setColor(color);
    StdDraw.fillOn();
    StdDraw.spot(2*radius);
}
```

(b) Write the `drawBullseye` method.

```java
public static void drawBullseye(double x, double y, double outerR, double ringThickness, Color color) {
    double radius = outerR;
    while(radius > 0){
        drawFilledCircle(x, y, radius, color);
        radius -= ringThickness;
        if(radius > 0) drawFilledCircle(x, y, radius, Color.white);
        radius -= ringThickness;
    }
}
```

Another valid solution is to have a loop and a counter keeping track of which color should be drawn. When the counter is even, draw a filled color circle. When the counter is odd, draw a white circle.
6. **TOY Programming (6 points)**

// register map: Add more if needed
// R[A] - a
// R[B] - b
// R[C] - temp

<table>
<thead>
<tr>
<th>Addr</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>8AFF</td>
<td>Read a from stdin</td>
</tr>
<tr>
<td>11</td>
<td>8BFF</td>
<td>Read b from stdin</td>
</tr>
<tr>
<td>12</td>
<td>CB1A</td>
<td>if (b==0) go output the gcd</td>
</tr>
<tr>
<td>14</td>
<td>D218</td>
<td>if a greater than b don't swap</td>
</tr>
<tr>
<td>15</td>
<td>1C0B</td>
<td>Swap: temp = b</td>
</tr>
<tr>
<td>16</td>
<td>1B0A</td>
<td>b = a</td>
</tr>
<tr>
<td>17</td>
<td>1A0C</td>
<td>a = temp</td>
</tr>
<tr>
<td>18</td>
<td>2AAB</td>
<td>a = a-b</td>
</tr>
<tr>
<td>19</td>
<td>C012</td>
<td>go to 12</td>
</tr>
<tr>
<td>1A</td>
<td>9AFF</td>
<td>Write a to stdout</td>
</tr>
<tr>
<td>1B</td>
<td>0000</td>
<td>halt</td>
</tr>
</tbody>
</table>

The swap requires the transfer of data between registers. There are many ways to move data between registers. Add R[0] was used in the solution above. XOR R[0] is another option. Storing the data in memory and then re-loading the registers is another (slightly less efficient) way. There are others as well.

7. **TOY Architecture (6 points)**

The following TOY instructions will still work:

- 0 Halt
- 7 Load address
- 8 Load
- A Load Indirect
- F Jump and link

Arithmetic and Logical operations (1, 2, 3, 4, 5, 6) will not work because they all need s to enter A Addr.

Store instructions (9, B) will not work because both need d to enter A Addr.

Of the Control instructions, the conditional branches (C, D) and jump register (E) will not work because they need d to enter A Addr.