1. **TOY Programming.**

   10: 8CFF \( R[C] \leftarrow \text{mem}[FF] \)
   11: 7101 \( R[1] \leftarrow 01 \)
   12: 7201 \( R[2] \leftarrow 01 \)
   13: 92FF write \( R[2] \) to stdout
   15: 2CC1 \( R[C] \leftarrow R[C] - R[1] \)
   16: DC13 if \((R[C] > 0) \) pc \(\leftarrow 13\)
   17: 0000 halt

   Alternatively, line 14 could be


2. **Scope.** Program prints the three lines:

   111
   2
   666

3. **Number Systems.**

   (a) \(-88\)
   (b) \(FFBF\)
   (c) \(2^{31}\) — All positive integers plus one more (for 0).
   (d) \(0\) — \(^\wedge\) denotes xor in Java, not exponents.

4. **Java Expressions.**

   (a) All expressions always evaluate to true.
   (b) Type conversion works as follows:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 * 0.2</td>
<td>double</td>
<td>2.2</td>
</tr>
<tr>
<td>(int) 11 * 0.2</td>
<td>double</td>
<td>2.2</td>
</tr>
<tr>
<td>11 * (int) 0.2</td>
<td>int</td>
<td>0</td>
</tr>
<tr>
<td>(int) (11 * 0.2)</td>
<td>int</td>
<td>2</td>
</tr>
</tbody>
</table>

   (c) i, ii, and iii are all equivalent.
5. **Arrays.** The three parts print out the following three lines:

```
0 0 1 1 2 2
0 1 2 3 4 5
5 4 3 2 1 0
```

6. **Recursive Methods.** We will accept either (d) the function is fine, or (e) with the explanation “call stack will overflow for large enough N” or “integer overflow.”

7. **Arrays and I/O.**

   (a) The program outputs the single line:
   ```
   aaa bbb ccc ccc bbb aaa
   ```

   (b) The pipe will consume the output from the first program, but has no effect on the second program because it uses command line arguments and not standard I/O, so the output is the single line:
   ```
   xxx yyy yyy xxx
   ```