1. First, we'll take a look at what exactly each of the subroutines does:
- *franz* calls *hanz* and increases *num* by adding 1 to its value
- *hanz* pops a number of times equal to the value of *num* (i.e. if *num* is 3, *hanz* will pop 3 times)

When the flag is clicked, the program starts by making *num* 1, then calls *franz* 3 times. In the table below, for each instruction we can see how *num* changes as the program is running.

We replaced the instruction
```
broadcast franz and wait
```
with the two instructions that form the subroutine:
```
broadcast hanz and wait
change num by 1
```

. Each time the “pop” sound happens, we added a * in the table in the 'pop' row.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>num</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>set num to 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>broadcast hanz and wait</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>change num by 1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>broadcast hanz and wait</td>
<td>2</td>
<td>**</td>
</tr>
<tr>
<td>change num by 1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>broadcast hanz and wait</td>
<td>3</td>
<td>***</td>
</tr>
<tr>
<td>change num by 1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

A. All we have to do is count the number of pops. In total, there will be **6 pops**.
B. *num* will be **4** at the end.

*Grading:* You got 8 points out of 10 for answering A correctly and *num* = 3 at the end. Points were split evenly between A and B. There was partial credit if you gave the wrong answer but explained how you solved the problems.

2. A “Parity-\(k\)” gate needs \(k-1\) “xor” gates. So a “Parity-16” gate will need 16-1=15 “xor” gates. 1 “xor” gate needs 2 *nots*, 1 *ors* and 2 *ands*, so 15 “xor” gates will need **30 nots**, **15 ors** and **30 ands**.

*Grading:* You got 7 out of 10 points for answering “32 nots, 16 ors and 32 ands”. Also, you got 7 out of 10 points for just adding all the gates together, which means you got 7 points for answering “75 gates”.

3. A. *E* = *notA* and *B*
B. *E* = (*notC* or *A*) and (*notA* and *B*)

*Grading:* You got 7 out of 10 if you didn't make all the reductions, for example, having 'E' in the final expression of E, instead of replacing E with what it actually is (from the previous steps). Another point was taken off for using foreign symbols (like “+”) in the expression of E
4.   A. Part 1:  notA or B  
     Part 2:  acc = notA  
                 acc = acc or B 
B. Part 1 : A and notB  
     Part 2 : acc = notB  
                 acc = acc and A 
C. Part 1 : (A or C) and notB  
     Part 2: acc = A  
                 acc = acc or C  
                 E = acc  
                 acc = notB  
                 acc = acc and E  
D. Part 1 : not(A and B)  
     Part 2 : acc = A  
                 acc = acc and B  
                 E = acc  
                 acc = not E 

**Grading:** Each of A, B, C, D were worth 2.5 points: 1 point for getting part 1 right (-0.5 for not using parentheses when needed), 1.5 points for getting the instructions right; for using illegal instructions, if the reasoning was correct, you got 0.5 points out of the 1.5 for the respective problem.