AP CSP Answer Key 1 – with sample full-credit responses

1) \(2^2 = 4\)  
(binary to decimal conversions below)

\[
\begin{align*}
0 0 &= 0, \\
0 1 &= 1, \\
1 0 &= 2, \\
1 1 &= 3
\end{align*}
\]

2) \(2^3 = 8\)

continued from above:

\[
\begin{align*}
0 0 0 &= 4, \\
0 0 1 &= 5, \\
0 1 0 &= 6, \\
0 1 1 &= 7
\end{align*}
\]

3) \(2^4 = 16\)

continued from the first two questions:

\[
\begin{align*}
1 0 0 0 &= 8, \\
1 0 0 1 &= 9, \\
1 0 1 0 &= 10, \\
1 0 1 1 &= 11, \\
1 1 0 0 &= 12, \\
1 1 0 1 &= 13, \\
1 1 1 0 &= 14, \\
1 1 1 1 &= 15
\end{align*}
\]

4) 4 bits

Since 3 bits can only represent all decimal numbers from 0 to 7, you’d need the fourth bit to represent the decimal numbers 8, 9 and 10.

5) 255

Since there are \(2^8 = 256\) possible values represented in 8 bits, those values must be 0 to 255, since they are consecutive and start at zero.

Alternatively, if you max out all 8 bits, in other words 1 1 1 1 1 1 1 1, you get \(1(1) + 1(2) + 1(4) + 1(8) + 1(16) + 1(32) + 1(64) + 1(128) = 255\), which is therefore the maximum value you can get using 8 bits (binary digits).

6) \(2^7 = 128\)

This number is important because a standard ASCII table goes from 0-127, meaning all 128 values on a standard ASCII table can be represented using 7 bits.

7) Because bits are usually represented in multiples of four, we put a leading 0 (or alternatively, what’s called a “parity” bit, which we have not discussed yet) before the seven bits that represent any alphabet letter. This makes 8 bits, or a byte.

Note: The extended ASCII table provides characters that need that 8th bit (in other words, the entire byte) in order to be represented.

8) \(2^32\) times as many values

32 bits can represent \(2^{32}\) possible values. 64 bits can represent \(2^{64}\) possible values. How many times does \(2^{32}\) go into \(2^{64}\)? Well \(2^{32} * 2^{32} = 2^{64}\). Or you can do \(2^{64}\) divided by \(2^{32}\) to find out.

9) V

First convert the hexadecimal 56 (five six, not fifty-six!) to decimal, which gives you \(5*16 + 6*1 = 86\) in decimal, which corresponds to the letter V using the given partial ASCII table.

10) Choice C

The original blurb says that the message itself is considered the data, whereas the timestamps and names behind all messages and comments are considered metadata. Choice C is the only choice that refers to information in the messages themselves and so is the choice that is most relevant to the data rather than the metadata.

11) Choice D
If you continue the pattern of each row, the next value for each row is 2000, 400, 400, and 10000, respectively. Which one is the highest? This answer would be the longest time for a company of 100,000 customers (the next heading in the pattern of headings).

12) The robot will initially move forward 3 times, since n = 3. Then the robot turns left 90 degrees and n goes down to 2 and the loop repeats. The robot moves forward 2 times, since n = 2. Then the robot turns left 90 degrees and n goes down to 1 and the loop repeats one last time (to complete a total of three iterations). On the third iteration, the robot moves forward 1 time, since n = 1. Then the robot turns left and n goes down to 0. The process ends because the loop has iterated 3 times, not because n = 0.

13) Displays 3 2
b takes on the value of a, which is 1. Then we add b and a, which is 1 + 1 = 2, and b takes on this value. Then we add b and a, which is 2 + 1 = 3 and a takes on this value.

14) This code will print out the maximum number in the list, namely 4. The loop goes through each item in the list, starting with the first number, 1. Is 1 greater than x (which is -1)? Yes, so x now equals 1. Is 0 greater than x (which is 1)? No, so x stays as 1. Is 4 greater than x (which is still 1)? Yes, so now x equals 4. Is 2 greater than x (which is 4)? No, so x stays as 4.
In fact, this code will display the maximum of any list, as long as there are numbers in the list that are greater than the initial value of x (which was -1). However, if all numbers in the list are below -1, then this algorithm will not display the maximum value of the list because no list value will be greater than -1, and so -1 will remain as the value of x. Therefore, it would be advantageous to make x equal to the first element in the list, namely x = list[1], to begin with. That way, each member of the list will definitely get a chance to compare to another member in the list rather than just comparing to -1.

15) Postponed until AP CSP Review 3

16) “Better late than never.”
Since onTime is false (as stated in the blurb), we do not display “Hello”. Instead, we follow the “Else” path, which first asks if “absent” is true, which it is not (as stated by the blurb), so the final “else” path is taken, which tells us to display “better late than never”.

17) This code would result in an infinite loop because even though i goes up to 2, it is immediately set back to 1 on the next iteration. You’d have to take out i = 1, in which case i would be allowed to progress upwards. Eventually, then it would get to 4 and the loop would be exited.
What would “sum” equal at that point? On the first iteration, sum + i = 0 + 0 (since i = 1 has now been taken out). So sum = 0. On the next iteration, sum + i = 0 + 1 (since i went up by one), so sum = 1. On the next iteration, sum + i = 1 + 2 (since i went up by 1), so sum = 3. On the next iteration, sum + i = 3 + 3 (since i went up by 1), so sum = 6. Now, if we went one more iteration, sum + i would equal 6 + 4 (since i went up by 1), but we’d never do that iteration because as soon as i went up to 4 at the end of the previous iteration, the loop would not repeat again. So the final value of sum is 6.

18) MoveAndTurn(2, 1), MoveAndTurn(4, 3), MoveAndTurn(2, 0)
The procedure MoveAndTurn takes two inputs, numMoves and numTurns. numMoves is the number of times the robot will go forward. numTurns is the number of times the robot will turn left. Notice that the robot goes forward a number of times and turns left a number of times before going forward another number of times and turning left another number of times (in the maze), so we will have to use MoveAndTurn multiple times, not just once.
The first time we use MoveAndTurn, the robot should go forward twice and turn left (by 90 degrees) once. The second time we use it, the robot should go forward four times and ideally turn right once (but since there is no code given for turning right, we must turn left 3 times since 3 left turns of 90 degrees equals one right turn of 90 degrees). Finally, the robot goes forward twice and doesn’t need to turn.