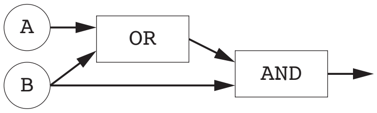
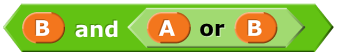
CS10 Fall 2016 Quest Answers

Question 1: Which is the *least correct* example of Abstraction, as we use in Computer Science? "Writing down a different smoothie recipe for every fruit, even though the only difference is the fruit itself" since that goes against the idea of Generalization, where you would generalize the recipe and have a single input, the fruit.

Question 2: What is the correct order of the numbers: 12_{16} (hex), 1110_2 (binary), 13_{10} (decimal)? $12_{16} = 1\times16^1 + 2\times16^0 = 16+2 = 18_{10}$. $1110_2 = 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 8 + 4 + 2 = 14_{10}$. So $13_{10} \leq 1110_2 \leq 12_{16}$.



Question 3: Which Boolean expression does model?

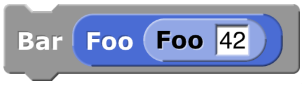


```
B and A or B
```

Question 4: If the output from the figure above is true, which can you say *for sure*? "B must be true", since the AND needs both of its inputs to be true, yet OR needs only one. So B has to be true, and the OR output has to be true (for the AND to be true), but if B is true, then A could be true or false and the OR output would still be true.

Question 5: Which of the following is a *false statement* about Algorithms? Given a particular problem, there is only one algorithm that can solve it. Proof by counterexample: doubling a number could be coded as n+n or n*2 or 3n-n or …

Question 6: Which of the following is the most powerful programming paradigm?
All are equally powerful, since all paradigms are Turing Complete.



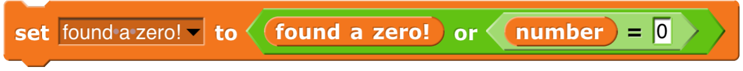
```
Bar Foo Foo 42
```

Question 7: Given the following error-free expression, what do you know *for sure* about the *Domain* and *Range* of `Foo`? The Domain of `Foo` contains the number 42? Yep, since it shows it in the example. The Range of `Foo` is exactly numbers? Nope, we know nothing about the range of `Foo`, except… The Domain of `Bar` *contains* the range of `Foo`? Yep! (that's actually all we know about the range of `Foo`) The Domain of `Bar` is *exactly* the range of `Foo`? Nope, who's to say that `Bar` couldn't take in other input in addition to whatever `Foo` gives us.

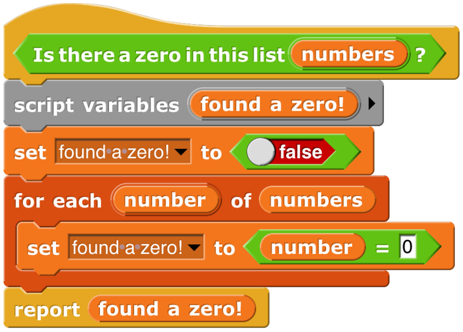
Question 8: The sprite starts in the center of the blank screen, and the user makes a call to `Draw Spiral`. We've zoomed in to the center to see the pretty pattern that was drawn. Since it takes 6 turns to face the same direction, it's 360/6 = 60 degrees.

Question 9: What does the program *always return*?
Since it always overrides "found a zero!" every time, ignoring what it was in the past, it always returns *true if there is a zero in the **last** element*.

Question 10: How could you fix the program so it works as intended?
We have to stop ignoring past values of "found a zero!", and since we want ANY of the zeros to trigger a return of true, not ALL, we say (in english) There's a zero in the list if there was one before THIS OR one is a zero.

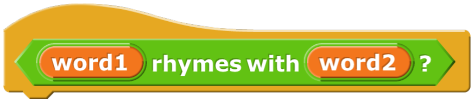


```
set found a zero! to found a zero! or number = 0
```



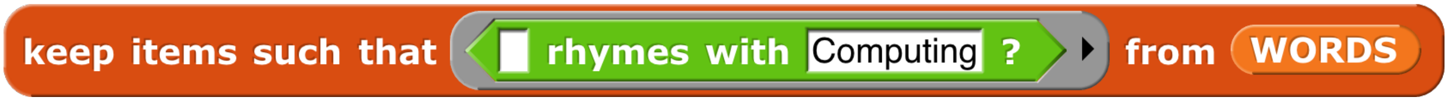
```
Is there a zero in this list numbers ?
script variables found a zero!
set found a zero! to false
for each number of numbers
    set found a zero! to number = 0
report found a zero!
```

Question 11: What is the worst-case running time of the algorithm? Linear, since a constant amount of work is happening with each iteration.

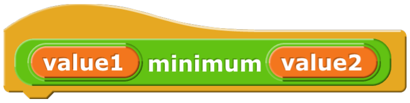


```
word1 rhymes with word2 ?
```

Question 12: Given a predicate function that determines whether two words rhyme, return all the words of a list that rhyme with *Computing*? That's a classic keep pattern, you're filtering all elements and only KEEPING the ones that satisfy a predicate, here rhyming.

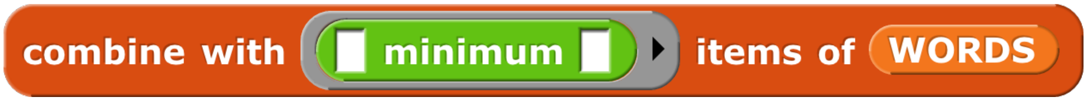


```
keep items such that ( rhymes with Computing ? ) from WORDS
```



```
value1 minimum value2
```

Question 13: Given a list of words and a function that determines the earlier of two values (words or numbers, etc.), return the word from the list that would be first when listed in alphabetical order? That's just combine, one of our classic examples.



```
combine with ( minimum ) items of WORDS
```