

## Quiz #3 Answers

1. We have discussed two ways of achieving repetitive behavior in Alice. What are they? Describe how they differ or when you would pick one over the other.

**The two ways we discussed for getting repetition were the for loop and the while loop. The for loop basically counts in Alice and we need to know how many times it should happen in advance. The while loop keeps going until some condition becomes false. For this reason the while loop works well when you don't know in advance how many times something needs to happen.**

**The doAllInOrder and doAllTogether constructs for lists are also methods of getting a single chunk of code to happen multiple times in Alice and while they weren't what I was originally considering, they are a valid answer to the way the question is asked and will be counted as such as long as you can differentiate them from the others. The fact that they only go through the contents of a list is a pretty good distinction.**

**Some people also mentioned just adding the code in multiple times. Again, this isn't what I was thinking of, but I have to admit it is a perfectly valid answer to the question as stated so it gets credit.**

2. In class we wrote code so that the mummy would go to one of our two characters, depending on who was closer. For only two characters this was straight forward. Now I want to do it for a list of them. How would you write a function that takes a list of "victim" objects and well as the single object of the "attacker" and returns the object from the list that is closest to the attacker. (Hint: this is going to use most of the constructs that we have talked about, but in many ways it is fairly simple. Just think of how you would do it if you had to use a tape measure because you couldn't just visually inspect the scene. Then translate that into Alice commands.)

**The basic idea here is to run through each item of the list, one at a time, and check to see if each one is closer to the "attacker" than those before it. Technically the best way to do this is to keep a variable that stores the closest one you have found so far, and the loop through the list and make the variable refer to the new one if it is closer than the one the variable is storing. When the loop is done, we simply return the variable because it will hold the object that was the closest one in the list.**

Extra Credit: Something about lists helps explain why the for loop in Alice starts counting at zero. What is it?

**The indexes of items in lists start at zero so when you loop through the contents of a list you want to start at zero as well.**