

|  |
| --- |
| REPORT TITLE  2018 |
|  |
| January 11  COMPANY NAME  Authored by: Your Name |

Logo

Name

# Title Heading

|  |
| --- |
| Subtitle Text Here To get started right away, just tap any placeholder text (such as this) and start typing to replace it with your own.  Want to insert a picture from your files or add a shape, text box, or table? You got it! On the Insert tab of the ribbon, just tap the option you need. |
| *“Find even more easy-to-use tools on the Insert tab, such as to add a hyperlink or insert a comment”* |
| To get started right away, just tap any placeholder text (such as this) and start typing to replace it with your own.  Want to insert a picture from your files or add a shape, text box, or table? You got it! On the Insert tab of the ribbon, just tap the option you need. |

## Fisher-Yates shuffle

I used a Fisher-Yates shuffle algorithm to display questions in random order to the Play-Only user. The algorithm do the following:

1. Gets a list of answers per question.
2. Gets the number of items in the list.
3. Starts counting backward from the number of items in the list.
4. Picks a random number from 1 to n (n being the total number of elements in the list A) and I call it number k.
5. Puts the item from count backward point n into the point k spot, and vice versa.
6. Repeat steps 3–5 until list n gets back to 0.

So let’s relate these steps to the code. On line 4 of the code, you see that the function takes a single input. It takes an IList<T>. I am assuming that not everyone is incredibly trained as a programmer, so even this could be a little confusing. What is an IList<T>? IList is a list that inherently has an index. Meaning that you can choose an item by its position in the list. This is useful as we said because we need to know how many items are left in list A. So what is <T>? This is a “generic” object. This means that this function can accept an indexed list of any type of object. So taking the long way to get there, line three is step 1.

Line 6 gets the number of items in the list, which is step 2. Line 7 starts the loop and says that it will run as long as n is greater than one, this is step 3. Line 8 immediately minuses one from variable n.

Organisation

Line 9 gets a random number inside the list of messages, that is counting down. Now, there are a lot of discussions online about true random numbers, and how to really get them. This solution makes use of the Random() function that .NET provides. In my answer, I made use of another piece of code I found on Stack Overflow [here](https://stackoverflow.com/questions/767999/random-number-generator-only-generating-one-random-number):

Sometime in the future, I’ll have to write an article about random. Just know that I used this instead of what is the solution random number. This is step 4.

Step 5 is the logic portion of the code. This logic is done by lines 10–12. Line 10 creates a generic object called value and places what is in the list at the random point in that object. Line 11 takes the item at the countdown point n and puts it in the random spot. Line 12 then makes the object that was put into the variable value in line 10 and puts it at point n on the list. This logic continues until you have counted n down to 1. When you are done, you have all the same items in the list but a different order.

With regards to Big O notation, this algorithm appears to me to be O(n). This notation means that the length of time it takes to complete this algorithm is directly proportional to the number of items in the list. If you increase the number of elements in the list, you increase the time the algorithm takes in a linear fashion.