Engineering Exercise

Instructions: The tasks below reflect some of the key roles and responsibilities the Senior Software Engineer will be asked to execute and perform. Please share your answers in the format that you feel would best communicate your work and rationale. If you have any questions along the way, please reach out to our recruitment team directly.

Programming

Container is a class with the looped double linked list of random bool values (see implementation below). It supplies an API to get or set the value of the current element, or move forward or backward in the list.

```
using System;

public class Container
{
   private class Node
   {
      public Node Next;
      public Node Prev;
      public bool Value;

      public Node(Node prev)
      {
        var randomGen = new Random(DateTime.Now.Millisecond);
         Value = randomGen.Next(2); < 1;
         Prev = prev;
      }
   }

   private Node current;

   public Container(int count = 0)
   {
      if (count < 1)
      {
         var randomGen = new Random(DateTime.Now.Millisecond);
         count = randomGen.Next(1, 9999); //Could be up to Int32.MaxValue, reduced for sake of test memory
      }

      Node prev = null;
      for (int i = 0; i < count; i++)
      {
         var currentNode = new Node(prev);
         if (prev != null)
         {
            prev.Next = currentNode;
         }
         if (current == null)
         {
            current = currentNode;
         }
         prev = currentNode;
        }
        prev.Next = current;
        current.Prev = prev;
    }

    public bool Value
    {
        get { return current.Value; }
        set { current.Value = value; }
    }

    public void MoveForward()
    {
        current = current.Next;
    }

    public void MoveBackward()
    {
        current = current.Prev;
    }
}
```

Task 1:

Write an algorithm in C# to find the number of nodes in the list. After the search, all values should be in their original state. Don't modify the Container class or use reflection; find an algorithmic way to do it using the API supplied by Container. Be aware of performance and memory allocation. Please comment your code extensively.

Task 2:

In Unity, create an Editor Window that displays an instance of the Container class as an infinitely scrollable list of true/false values. Include a button in the window that creates a new instance of Container with random node count to serve as the data source for the scrollable list. The visualization should be robust to resizing the window. It's fine to turn in just the code; we don't need the whole Unity project.

Architecture

Imagine that we are building a single-player game in which the local player controls their avatar in the foreground in real time. Our Design team has requested that we implement a feature called "Phantasms." The Phantasm feature requires us to display in the background other AI agents called "phantasms" that have simplified movesets that are a subset of what the local player can do. Even though they are AI agents, the actions of the phantasms should be derived from other human players currently playing the game, with a single phantasm representing a single real human player. The phantasms don't affect gameplay, and thus do not need to be closely controlled in real time like a true multiplayer game, but we would like them to update their behavior parameters reasonably often as other human players make moves (about every minute or so) and only represent real players currently playing the game. Assume that we have a robust server architecture that the local client can query to learn the current state of other players via a pre-existing middle layer in the client, but that the server tech stack does not support sockets or any other data pushing technology. Also assume that we have animated avatars that can accept commands to perform individual actions from the simplified phantasm moveset.

Task 3:

Imagine that you are responsible for implementing the Phantasms client feature, in collaboration with a server team. Please describe at a high level how you would design the systems to support this feature, aiming for an architecture that would easily accommodate future feature extension or iteration. Write a set of classes in C# pseudocode that would fulfill your design. Feel free to append diagrams that you feel would help explain the architecture.