CSC 242
Lab 3

Option 1:
Define a function **makeTwoWay** that expects a singly linked structure as its argument. The function builds and returns a doubly linked structure that contains the items in the singly linked structure. (Note: The function should not alter the argument's structure.)

Option 2:
Add the methods **insert** and **remove** to the **Array** class. These methods should use the strategies discussed in this chapter (look for **Operations on Arrays**), including adjusting the length of the array, if necessary.

The **insert** method expects a position and an item as arguments and inserts the item at the given position. If the position is greater than or equal to the array's logical size, the method inserts the item after the last position currently occupied in the array. **insert** may require the **Array** to increase in size (use the doubling approach).
Observe the following scenarios describing the insert behavior:
Scenario 1.

```
arr: [4, 2, None, None, None]
arr.insert(4, 8)
```

Since position 4 is greater than or equal to the array's logical size (2), the 8 is inserted right after the 2. The 2 was in the last position occupied.

```
arr: [4, 2, 8, None, None]
```

Scenario 2.

```
arr: [4, 2, 8, 1, 7]
arr.insert(4, 10)
arr: [4, 2, 8, 1, 10, 7, None, None, None, None]
```

Since position 4 was occupied by the 7, the 7 gets shifted up one position to make room for the 10. Because there wasn't enough room to store the 10 into the array, the array's physical size (5) was increased to 10.

The **remove** method expects a position as an argument and removes and returns the item at that position. The **remove** method's precondition is $0 \leq index < size()$. The **remove** method should also set the vacated array cell to the fill value. This vacated array cell is the last cell that was shifted up in the array to adjust for the removed position. **remove** may require the **Array** to decrease in size (use the strategy proposed in the chapter readings).
Here is a snapshot of the **remove** behavior:

```
arr: [4, 2, 8, 6, None]
arr.remove(1)
arr: [4, 8, 6, None, None]
```

The highlighted position represents the vacated cell that was set to None.

Add the method **__eq__** to the **Array** class. Python runs this method when an **Array** object appears as the left operand of the **==** operator. The method returns *True* if its argument is also an **Array**, it has the same logical size as the left operand, and the pair of items at each *logical* position in the two arrays are equal. Otherwise, the method returns *False*.

Test all three of these functions sufficiently in a driver program of your choosing.

Question:
Using the strategy proposed to decrease the size of the Array, what is the load factor required for this condition to be satisfied?