Binary Circles

Function: CheckForDistinct(mylist)

Input: mylist = The list that is tested to see whether its elements are distinct or not

Output: mylist has distinct values, mylist doesn't have distinct values

```
For elements in mylist
    If that element appears more than once in the list
        Return a statement saying that mylist does not have distinct values and
        discontinue the for loop since that element appears more than once in the list
If the for loop runs without any breaks, then that means no elements were repeated and a
statement saying that mylist has distinct values is returned
```

Main Code:

Variable: N = The number that 2 will be raised to in order to determine circle length

Variable: CircleSize = The length of the circle equal to 2^N

Variable: CircleIndicator = A list that will consist of N elements, all of them zero

```
For zeros going from 0 to N
    Add a zero to the list CircleIndicator
```

Variable: Circles = A list that will store all circles whose first N elements are all zeros

Variable: UniqueCircles = A list that will store all circles with distinct N-digit subsequences

```
Enter a value for N that is greater than 1
While the number of circles in Circles is less than the maximum number of circles that can have
their first N elements be zero
    Variable: CircleArray = A copy of CircleIndicator since we want the circles' first N
    elements to be zeros
    For i going from N to CircleSize + (N - 1)
        Randomly add either a "1" or "0" to CircleArray
    For ii going from 0 to N-1
        Equate any elements of CircleArray with index ii + CircleSize to elements with
        index ii to represent an overlapping circle
    If CircleArray is not already in the list Circles
```

```
        Add CircleArray to Circles (this way, all circles in Circles are distinct)
For j going from index 0 to the largest index of Circles
    Variable: templist1 = The jth circle in Circles
    Variable: BitArray = A list that will store all N-digit subsequences in a circle
    For jj going from 0 to CircleSize
        Variable: bit = A string that will represent a N-digit subsequence in a circle
        For bitparts going from jj to jj + N
            Concatenate the bitparts-th element of the circle (templist1) to bit
        Store that bit into BitArray
    If all N-digit subsequences in BitArray are distinct
        Save the jth circle with distinct N-digit subsequences to UniqueCircles
Variable: UniqueSum = An integer that will store the sum of all circles with distinct N-digit subsequences expressed as numbers
For k going from index 0 to the maximum index of UniqueCircles
    Variable: CircleString = A string that will store all elements of a circle
    Variable: templist2 = The kth circle in UniqueCircles
For kk going from 0 to CircleSize (since the last N – 1 elements in templist2 are supposed to be the beginning of the circle)
        Concatenate each element of templist2 into CircleString
    Convert CircleString into an integer and add it to UniqueSum
Print out UniqueSum with appropriate descriptive text
```