Exercise 3 9/30/22

4.

- Only 1 VIP can exist in the group at once because the VIP is someone who is known by everyone in the room but doesn't know anyone. If there were two or more that means they all wouldn't know eachother, also meaning that everyone in the room don't know them

- This could take $O(n^2)$ because we can iterate through every indiudual and see who they know and who knows them. To go down and compare every relationship for one person would be n times and for n people in the room it sums up to $O(n^2)$.

- Algorithm: VIP

```
1. cont=0
2. for int i=0 to n
3.   for int j =0 to n
         if ( j knows i and i doesnt know j)
            Count++;
         end if
         if (Count=n)
            return i;
         end if
      end for
      count=0;
   end for
```

This algorith can also take O(n) if we do it a more recursive way. If we recursively go through with the dataset w/ a stack comparing if A knows B, because if A knows B then A cannot be the celebrity. But B can. And vice versa if B knows A. The potential VIP is the last one standing. So this only require one iteration through n.

Algorithm: FastVIP

```
1. Stack st = new Stack<>();
2. for loop to push everyone to Stack {}
3. Int c = 0;
   While (st.size() > 1)
       int i = st.pop()
       int j = st.pop()
       if (i knows j)
           st.push(i)
       else
           st.push(j)
       end if
   end while
   c = st.pop

   for loop through all elements
       If (k != c & c knows k | k doesn't know c)
           return false;
   end for
   return true;
```

2. This algorithm can be O(n) time complexity depending on how you're searching. In my approach we start from the top right corner.
From there, there are three possible cases; the number we are searching for is less than, greater than or equal to. If it is greater than we know all elements in the current row are less than it, so we can discard the whole row. Same with if it is less than, that means all elements in the column are greater than it and we can discard the whole column. Than we can continue to do this until we find the element.

Algorithm: search (int [][] A, int s)

```
1. int i=0, j=n-1
2. while (i<n & j>=0)
3.    if (A [i][j] = s
      end if
      if (A [i][j] > x)
          j--;
      else
          i++;
      end if
   end while
```

5. Don Knuth has been called the "father" of the analysis of algorithms. He is a big part of the fundamental contributions of theoretical computer science. Contributing to the development of, and the creation of formal mathematical techniques for the computational complexity of algorithms. In addition Knuth is the creator of the TeX typesetting system and also created the WEB/CWEB computer programming systems.

1. The Big-Oh estimate of this code is $O(n^2)$ this is because we are loping through the nested for loops n^2 amount of times then when j is reduced by 2 each time it adds $O(\log n)$ to the time complexity but since $O(n^2)$ is the dominant term we stick with $O(n^2)$

3. Lets suppose we have a person A. And suppose set $\{B, C, D\}$ are all friends with A, if there are a pair of friends in the set then $\{A, B, C\}$ is a set of mutual friends. Otherwise $\{B, C, D\}$ are mutual strangers friends. And this case holds in vice versa if A is a stranger to set $\{B, C, D\}$