Period: 2
Team Name: **Toot Toot Cabbage**
Roster: **Bill Ni, Andrew Shao, Woobin Peco**

Pseudocode explaination

```
boolean searchMatrix(int[][] matrix, int desiredValue) {
    int currentRow = 0
    int currentColumn = matrix.length - 1

    while (currentColumn >= 0 && currentRow < matrix.length) {

          if (matrix[currentRow][currentColumn] == desiredValue)
               return true //this checks if the current element is what we want

          else if (matrix[currentRow][currentColumn] < desiredValue)
               //if the currentElement is less than what we want
               currentRow++ //we increment the row, because it cannot be in that row

          else //only other possible case is if the element is greater than what we want
               currentColumn--
     }
    return false;
}
```

Rundown of the algorithm:
We start off at the top-right most element of the matrix, and then repeat the following tests

- If the current element is the desired value, return true

- If the current element is less than the desired value, go to the next row
    - This is because, as we are starting in the last element of each row, that means we are at the largest element of each row. If the largest element of each row is still less than the desired value, that means there is absolutely no possibility of it being in that row, so we can continue down to the next row.
    - This still holds true, even if we are not at the largest element, as there is still a possibility it is in the following row, so we continue down the rows.

- If the current element is greater than the desired value, go to the column on the left
    - This is because, as we are starting in the last element of each row, we are at the largest element, and if the largest element is greater than the desired value, then there is a possibility that the desired value is in the current row and we can search the row by decrementing the current column value.
    - This still holds true, even if we are not at the largest element of the row, as we will always decrement the current column value until we reach our value, the current element is less than the desired value, or current column reaches a value of -1

Finally, if we break out of the loop, that means the value was NOT found, and thus, we return false.

Big O Analysis:
This will run in O(2n - 1) time, however, that is the same as O(n) because as n increases, the time to run the function increases LINEARLY.

O(2n) is the same as O(n) because in both cases, multiplying n by 3 triples the amount of time it takes for the function to run, thus making it a linear relationship.

The “- 2” does not affect it whatsoever either, as it is just a constant.

Thus, this algorithm runs in O(n) time