Pseudo code for LockerPuzzle:

1. Create a Boolean array with 101 items. 101 because the first locker is 1, and the last locker is 101, making 100 lockers.
2. Make a for loop that goes through every locker excluding 0
   * Make the value of the lockers true, or open.
3. Make another for loop starting with the second person going up until the hundredth student.
   * Close every jth locker.
4. Make another for loop that goes through every locker
   * If the locker is open, then print that the locker is open

Pseudo code for PatternRecognition:

1. In the main method:
   * Create variable rows to store the amount of rows
   * Create variable columns to store the amount of columns
   * Prompt the user to input several rows
   * Record the number of rows in variable rows
   * Prompt the user for the number of columns
   * Record the number of columns in variable columns
   * Create 2-dimensional array with the first length being rows, and the second length being columns.
   * Prompt the user to enter data, 1 row per line.
   * Use a for loop to go through the every row
     1. Use another for loop to read every number in a column.
        1. Record the next number in the next column, if none remain, put in the first of the next row until there are no more rows or numbers available
   * Create a Boolean to hold if the 2-dimensional array has consecutive fours, calling isConsecutiveFour and passing the 2-dimensional array as an argument.
   * Display the answer to the user
2. In the isConsecutiveFour method, takes 1 argument, a 2-dimensional int
   * Create a variable columnLength and set to the column length of the 2-D array
   * Create a variable rowLength and set to the row length of the 2-D array.
   * Check if there are vertical consecutive four:
     1. Make a for loop that goes through every column
        1. Set the starting number to the first value in the ith column.
        2. Set a counter to 1.
        3. Loop through every row
           1. Check if the starting number matches the number in the row,

If it does, add one to the counter

If it doesn’t, make that the starting number and reset the counter, go again

* + - * 1. Check if the counter goes to 4

If it does, return true as there is a consecutive four

* + Check if there is a horizontal consecutive four
    1. Make a loop that goes through every row
       1. Set the starting value to the first item in the ith row.
       2. Set a counter to 1.
       3. Make a loop that goes through every column
          1. Check if the starting number is equal to the ith row in and the jth column.

If it is, add 1 to the counter

If it’s not, set the starting value to the ith row and jth column and reset the counter to 1.

* + - * 1. Check if the counter is 4

If it is, return true

* + Check for diagonal consecutive four with the row, left to right, top to bottom
    1. Make a loop that goes through every row
       1. Set a starting number to the first item in the ith row.
       2. Set a counter to 1
       3. Make a loop that checks the next row and next column, make sure the it doesn’t go above the length of the row and column or you’ll get an out of bounds error
          1. Check if the value in the kth column in the jth row is equal to the starting number.

If it is, add 1 to the counter

If it’s not, set the starting number to the kth row of the jth column and reset the counter to 1.

* + - * 1. Check if the counter is equal to 4

If it is, return true

* + Check for diagonal consecutive four with the column, left to right, top to bottom
    1. Make a loop that goes through every column
       1. Set a starting value to the ith item in the first row.
       2. Set a counter to 1
       3. Make a loop that checks the next column and row, make sure the it doesn’t go above the length of the row and column or you’ll get an out of bounds error
          1. Check if the starting number is equal to the jth column in the kth row.

If it is, add 1 to the counter

If it’s not, set the starting number to jth column in the kth row.

* + - * 1. Check if the counter is 4

If it is, return true

* + Check for diagonal consecutive four with the column, right to left, top to bottom
    1. Make a loop that goes through every column
       1. Set the starting value to the ith item in the first row.
       2. Set a counter to 1
       3. Make a loop that that checks the next column and row to the left, and make sure that you don’t exceed the row length, so you don’t get an out of bounds error
          1. Check if the starting number is equal to the kth column in the jth row.

If it is, add 1 to the counter

If it’s not, reset counter and set the starting value to the kth column in the jth row.

* + - * 1. Check if the counter is 4

If it is, return true

* + Check for a diagonal consecutive four with the row, right to left top to bottom
    1. Make a loop that goes through every row
       1. Set the starting value to the last value in the ith row.
       2. Set a counter to 1
       3. Make a loop that checks the left column and bottom row from the first value, make k go down every time
          1. Check if the starting number matches the value in the kth column of the jth row.

If it does, add 1 to the counter

If it doesn’t, reset the counter and set the starting value to the current value

* + - * 1. Check if the counter has reached four

If it has, return true

* + If all loops have gone through and got nothing, return false.

Test plan for LockerPuzzle:

I searched online what the program is supposed to output and checked it with the output of the program, and they matched. Screenshot is below, there’s not much else needed to check the program, it just needs to pass this test.

Screenshot:

A screenshot of a cell phone

Description automatically generated

Test plan for PatternRecognition:

I will put in the examples given in the instructions and make sure those pass true, and then check another set of data that doesn’t have consecutive fours, should return false.

Test cases for PatternRecognition:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cases** | **Input** | **Expected Result** | **Actual Result** | **Did it pass?** |
| Case 1 | Enter the number of rows: 6  Enter the number of columns: 7  Data:  0 1 0 3 1 6 1  0 1 6 8 6 0 1  5 6 2 1 8 2 9  6 5 6 1 1 9 1  1 3 6 1 4 0 7  3 3 3 3 4 0 7 | true: There are consecutive fours. | true: There are consecutive fours. | Y |
| Case 2 | Enter the number of rows: 6  Enter the number of columns: 7  Data:  0 1 0 3 1 6 1  0 1 6 8 6 0 1  5 6 2 1 6 2 9  6 5 6 6 1 9 1  1 3 6 1 4 0 7  3 6 3 3 4 0 7 | true: There are consecutive fours. | true: There are consecutive fours. | Y |
| Case 3 | Enter the number of rows: 6  Enter the number of columns: 7  Data:  0 1 0 3 1 6 1  0 1 6 8 6 0 1  5 5 2 1 8 2 9  6 5 6 1 1 9 1  1 5 6 1 4 0 7  3 5 3 3 4 0 7 | true: There are consecutive fours. | true: There are consecutive fours. | Y |
| Case 4 | Enter the number of rows: 6  Enter the number of columns: 7  Data:  0 1 0 3 1 6 1  0 1 6 8 6 0 1  9 6 2 1 8 2 9  6 9 6 1 1 9 1  1 3 9 1 4 0 7  3 3 3 9 4 0 7 | true: There are consecutive fours. | true: There are consecutive fours. | Y |
| Case 5 | Enter the number of rows: 6  Enter the number of columns: 7  Data:  0 1 0 3 1 6 1  0 1 6 8 6 0 1  5 6 2 1 8 2 9  6 5 6 1 1 9 1  1 3 6 1 4 0 7  3 4 3 3 3 0 7 | false: There aren’t consecutive fours. | false: There arent consecutive fours. | Y |

Screenshot for Case 1:

A screenshot of a social media post

Description automatically generated

Screenshot for Case 2:

A screenshot of a social media post

Description automatically generated

Screenshot for Case 3:

A screenshot of a social media post

Description automatically generated  
Screenshot for Case 4:

A screenshot of a social media post

Description automatically generated

Screenshot for Case 5:

A screenshot of a cell phone

Description automatically generated

Flowchart:

I believe that the Pseudo code/Algorithm is enough to not warrant a flowchart for either project.

UML Class Diagrams:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Lessons Learned:

I learned how to work regular arrays and multidimensional arrays. I learned that there are many ways to loop through arrays and that it’s possible to loop through them from right to left without reversing the whole thing first, by doing the length -1 and making the index go down as opposed to making the index go up.

Checklist for LockerPuzzle:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Source java files** | **Y** |  |
|  | **Compressed files:** | **Y** |  |
|  | FirstInitialLastName\_Project5\_1Moss.zip | **Y** |  |
|  | FirstInitialLastName\_Project5\_2Moss.zip | **Y** |  |
|  | FirstInitialLastName\_Project5\_doc.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs** | **Y** |  |
|  | **Checklist is completed and included in the Documentation** | **Y** |  |
|  | **Documentation file:** | **Y** |  |
|  | **Comprehensive Test Plan** | **Y** |  |
|  | **Screenshots based on Test Plan** | **Y** |  |
|  | **UML Diagram** | **Y** |  |
|  | **Algorithms/Pseudocode** | **Y** |  |
|  | **Flowchart** | **Y** |  |
|  | **Lessons Learned** | **Y** |  |

Checklist for PatternRecognition:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Source java files** | **Y** |  |
|  | **Compressed files:** | **Y** |  |
|  | FirstInitialLastName\_Project5\_1Moss.zip | **Y** |  |
|  | FirstInitialLastName\_Project5\_2Moss.zip | **Y** |  |
|  | FirstInitialLastName\_Project5\_doc.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs** | **Y** |  |
|  | **Checklist is completed and included in the Documentation** | **Y** |  |
|  | **Documentation file:** | **Y** |  |
|  | **Comprehensive Test Plan** | **Y** |  |
|  | **Screenshots based on Test Plan** | **Y** |  |
|  | **UML Diagram** | **Y** |  |
|  | **Algorithms/Pseudocode** | **Y** |  |
|  | **Flowchart** | **Y** |  |
|  | **Lessons Learned** | **Y** |  |