COMP 322 Worksheet 21: Eureka-style Speculative Parallelism.

Name: _____ **Net Id:** _____

The code snippet below performs a eureka-style search on a 2-D array with a fixed number of tasks. Each task uses the `next()` operation to ensure the computation progresses in a lock step manner where all tasks execute one call to `check()` before performing the equality (==) comparison. What does the program print when it completes execution? Also, print the number of == comparisons performed by the program. Remember that once the search eureka has been resolved (via a call to `offer()`), subsequent calls to `check()` will cause the task to terminate.

```
final int numRows = 10;
final int numCols = 100;
final int[][] dataArray = new int[numRows][numCols];
for (int i = 0; i < numRows; i++) {
    for (int j = 0; j < numCols; j++) {
        dataArray[i][j] = 100 * i + j;
    }
}

final int searchElement = 625;
final HjSearchEureka<int[]> eureka = newSearchEureka();
finish(eureka, ()-> {
      forasyncPhased(0, numRows - 1, (i) -> {
        for (int j = 0; j < numCols; j++) {
            final int[] elemIndex = {i, j};
            eureka.check(elemIndex);
            next(); // barrier
            if (dataArray[i][j] == searchElement) {
                eureka.offer(elemIndex);
            }
            next(); // barrier
         } // for
      }); // forasyncPhased
}); // finish
final int[] index = eureka.get();
System.out.println("Result = " + Arrays.toString(index));
```

Answer:

Result: _____

Number of == comparisons: _____