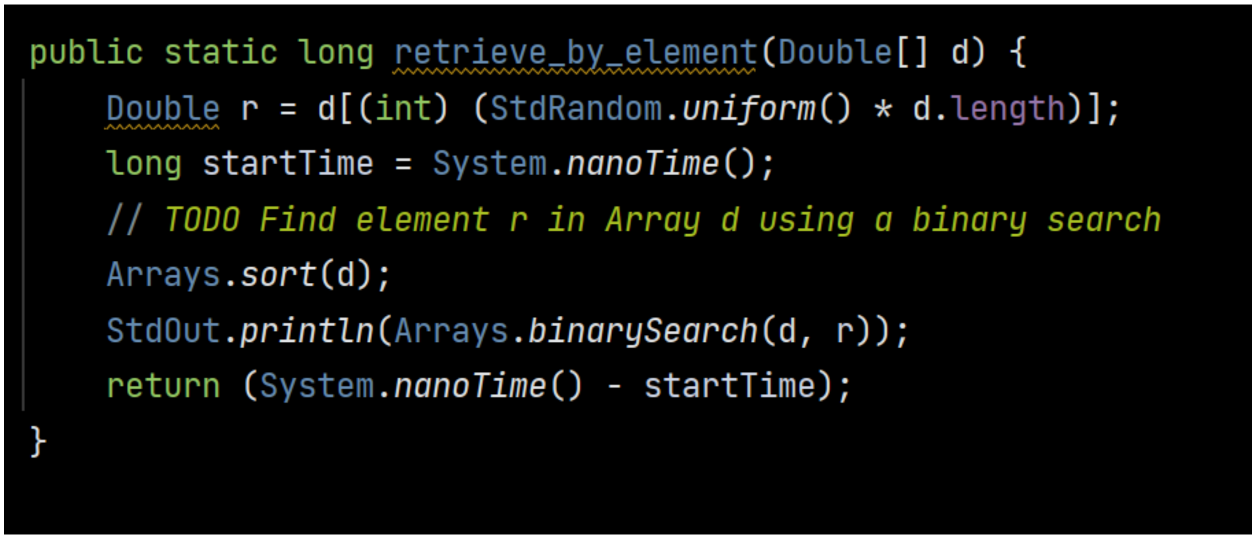
Task 2



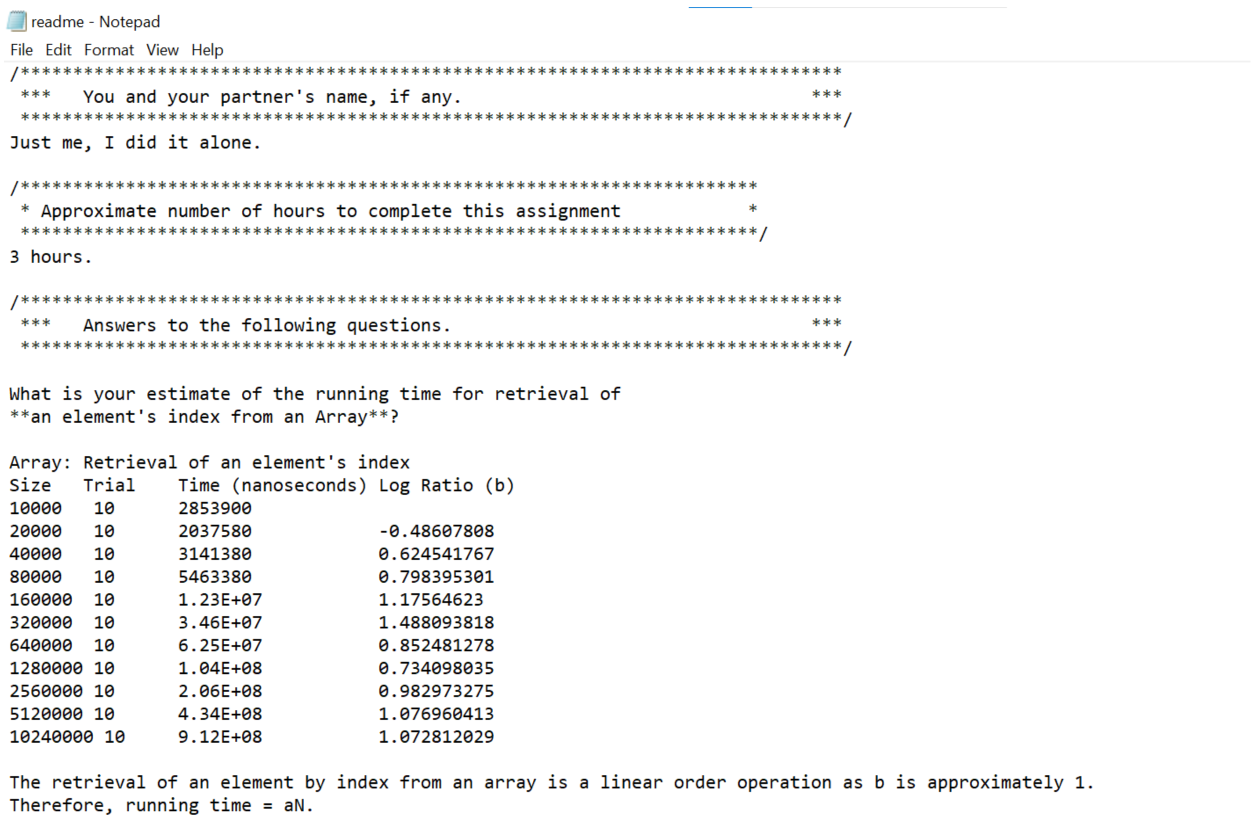
```java
public static long retrieve_by_element(Double[] d) {
    Double r = d[(int) (StdRandom.uniform() * d.length)];
    long startTime = System.nanoTime();
    // TODO Find element r in Array d using a binary search
    Arrays.sort(d);
    StdOut.println(Arrays.binarySearch(d, r));
    return (System.nanoTime() - startTime);
}
```

Task 3

```java
import java.util.LinkedList;


public class LinkedListExperiments {

  public static LinkedList<Double> initialize(int size) {

    // A LinkedList of random doubles

    LinkedList<Double> d = new LinkedList<Double>();

    for (int i = 0; i < size; i++) {

      d.addLast(StdRandom.uniform()); // add double into the LinkedList.

    }

    return d;

  }


  public static long retrieve_by_index(LinkedList<Double> d) {

    int i = (int) (StdRandom.uniform() * d.size()); // get a random index within the list bound.
    long startTime = System.nanoTime();

    double r = d.get(i);

    return (System.nanoTime() - startTime);

}


public static long retrieve_by_element(LinkedList<Double> d) {

    int i = (int) (StdRandom.uniform() * d.size());

    double r = d.get(i);

    long startTime = System.nanoTime();

    StdOut.println("index: " + d.indexOf(r));

    return (System.nanoTime() - startTime);

}


public static void main(String[] args) {

    int size = Integer.parseInt(args[0]);

    long trials = Long.parseLong(args[1]);


    LinkedList<Double> d = initialize(size);

    /* System.out.println("Timing retrieval by index...");

    long index_retrieval = 0;

    for (long t = 0; t < trials; t++) {

        index_retrieval += retrieve_by_index(d);

    }

    System.out.println(

            "Retrieval by index: " + (index_retrieval / ((double) trials))
```

```
                + " nanoseconds on average");


    System.out.println("Sorting array...");

    d.sort(Comparator.naturalOrder());

    */


    System.out.println("Timing retrieval by element...");

    long element_retrieval = 0;

    for (long t = 0; t < trials; t++) {

      element_retrieval += retrieve_by_element(d);

    }

    System.out.println(

        "Retrieval by element: " + element_retrieval / ((double) trials)

            + " nanoseconds on average");



  }

}
```

Task 4



readme - Notepad

File Edit Format View Help

```
/*******************************************************************************
 ***   You and your partner's name, if any.                                 ***
 *******************************************************************************/
Just me, I did it alone.

/***********************************************************************
 * Approximate number of hours to complete this assignment            *
 ***********************************************************************/
3 hours.

/*******************************************************************************
 ***   Answers to the following questions.                                  ***
 *******************************************************************************/

What is your estimate of the running time for retrieval of
**an element's index from an Array**?

Array: Retrieval of an element's index
Size    Trial    Time (nanoseconds) Log Ratio (b)
10000    10      2853900
20000    10      2037580            -0.48607808
40000    10      3141380            0.624541767
80000    10      5463380            0.798395301
160000   10      1.23E+07           1.17564623
320000   10      3.46E+07           1.488093818
640000   10      6.25E+07           0.852481278
1280000 10       1.04E+08           0.734098035
2560000 10       2.06E+08           0.982973275
5120000 10       4.34E+08           1.076960413
10240000 10      9.12E+08           1.072812029

The retrieval of an element by index from an array is a linear order operation as b is approximately 1.
Therefore, running time = aN.
```

```
What is your estimate of the running time for retrieval of
**an element by index from a LinkedList**?

Here are my results for the retrieval of an element by index in LinkedList using 10 trials. I found that the
retrieval of an element by index is a linear order operation.

Since b = 1, therefore: Running time = aN

LinkedList: Retrieval of an element by index
Size    Trial    Time (nanoseconds)      Log Ratio (b)
10000    10      48900
20000    10      78660                   0.69
40000    10      134560                  0.77
80000    10      168940                  0.33
160000   10      285680                  0.76
320000   10      512590                  0.84
640000   10      969440                  0.92
1280000 10       1863240                 0.94
2560000 10       4877420                 1.39
5120000 10       9755180                 1.00
10240000 10      1.95E+07                1.00
```

What is your estimate of the running time for retrieval of
an element's index from a LinkedList?

Same as the previous two questions, the retrieval of an element's index from a LinkedList is a linear order operation. b is approximately 1. Therefore running time = aN.

```
LinkedList: Retrieval of an element's index
Size    Trial    Time (nanoseconds) Log Ratio (b)
10000    10      631650
20000    10      611080              -0.05
40000    10      944630              0.63
80000    10      898380             -0.07
160000   10      1178980             0.39
320000   10      1479370             0.33
640000   10      3241120             1.13
1280000 10       6061230             0.90
2560000 10       1.17E+07            0.95
5120000 10       1.90E+07            0.70
10240000 10      4.58E+07            1.27
```

```
/*****************************************************************************
 ***   Do you attest that this work is your own, in accordance with the     ***
 ***   statement on academic integrity in the syllabus?                     ***
 *****************************************************************************/
```

Yes or no?
Yes

```
/*****************************************************************************
 ***   List any other comments here.                                        ***
 *****************************************************************************/
```