

```

import java.util.*;
class Lab2{
    public static void main(String [] args){
        int insVal;
        int TS;
        Random MyRand = new Random();
        double[] loadsize = {.1, .2, .3, .4, .5, .6, .7, .8, .9, 1.0};
        int[] ANOLPFT = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //average number of linear probes for a trial
        int[] MNOLPFT = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //maximum number of linear probes for a trial
        int[] ANOLPFL = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //average number of linear probes for a load
        int[] MNOLPFL = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //maximum number of linear probes for a load
        int[] ANOQPFT = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //average number of quadratic probes for a trial
        int[] MNOQPFT = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //maximum number of quadratic probes for a trial
        int[] ANOQPFL = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //average number of quadratic probes for a load
        int[] MNOQPFL = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; //maximum number of quadratic probes for a load

////////////////////////////////////Begin Linear Table Run
////////////////////////////////////
        System.out.println("Linear Hash Tables");
        for(int i = 0 ; i < 10 ; i++) //will run 10 linear Hash Tables of different load values
        {

            TS = nextPrime((int)(Math.round (10000 / loadsize[i])) );
            LinearProbingHashTable LTable = new LinearProbingHashTable(TS);
            for(int j = 0 ; j < 10 ; j++) //will run 10 times for good averages
            {
                while(loadsize[i] > LTable.getLoad()) //fills until load is reached
                {
                    insVal = (MyRand.nextInt()%500000);
                    LTable.insert(insVal);
                }

                ANOLPFT[j] = LTable.TotalProbes/10000;
                MNOLPFT[j] = LTable.MaxProbes;
            }
            for(int k = 0 ; k < 10 ; k++){
                if(MNOLPFL[i] < MNOLPFT[k])
                    MNOLPFL[i] = MNOLPFT[k];
            }
            ANOLPFL[i] = ANOLPFT[0] + ANOLPFT[1] + ANOLPFT[2] + ANOLPFT[3] + ANOLPFT[4] +
                ANOLPFT[5] +
                ANOLPFT[6] + ANOLPFT[7] + ANOLPFT[8] + ANOLPFT[9];
            ANOLPFL[i] = ANOLPFL[i]/10;
            System.out.println("For a load of " + loadsize[i] + " average probes is: " + ANOLPFL
                [i]);
            System.out.println("For a load of " + loadsize[i] + " maximum probes is: " + MNOLPFL
                [i]);
        }

////////////////////////////////////Begin Quadratic Table Run
////////////////////////////////////
        System.out.println("");
        System.out.println("Quadratic Hash Tables");
        for(int i = 0 ; i < 10 ; i++) //will run 10 linear Hash Tables of different load values
        {

            TS = nextPrime((int)(Math.round (10000 / loadsize[i])) );
            QuadraticProbingHashTable QTable = new QuadraticProbingHashTable(TS);
            for(int j = 0 ; j < 10 ; j++) //will run 10 times for good averages
            {
                while(loadsize[i] > QTable.getLoad()) //fills until load is reached
                {
                    insVal = (MyRand.nextInt()%500000);
                    QTable.insert(insVal);
                }

                ANOQPFT[j] = QTable.TotalProbes/10000;
                MNOQPFT[j] = QTable.MaxProbes;
            }
            for(int k = 0 ; k < 10 ; k++){

```

```

        if(MNOQPFL[i] < MNOQPFT[k])
            MNOQPFL[i] = MNOQPFT[k];
    }
    ANOQPFL[i] = ANOQPFT[0] + ANOQPFT[1] + ANOQPFT[2] + ANOQPFT[3] + ANOQPFT[4] +
        ANOQPFT[5] +
        ANOQPFT[6] + ANOQPFT[7] + ANOQPFT[8] + ANOQPFT[9];
    ANOQPFL[i] = ANOQPFL[i]/10;
    System.out.println("For a load of " + loadsize[i] + " average probes is: " + ANOQPFL
[i]);
    System.out.println("For a load of " + loadsize[i] + " maximum probes is: " + MNOQPFL
[i]);
}

////////////////////////////////////begin final calculations
////////////////////////////////////
System.out.println("");
System.out.println("Overall Values");
int AveLinProbes = ANOLPFL[0] + ANOLPFL[1] + ANOLPFL[2] + ANOLPFL[3] + ANOLPFL[4] +
    ANOLPFL[5] +
    ANOLPFL[6] + ANOLPFL[7] + ANOLPFL[8] + ANOLPFL[9];
int MaxLinProbes = 0;
for(int a = 0 ; a < 10 ; a++){
    if( MaxLinProbes < ANOLPFL[a])
        MaxLinProbes = ANOLPFL[a];
}
System.out.println("Average Number of Linear Probes: " + AveLinProbes);
System.out.println("Maximum Number of Linear Probes: " + MaxLinProbes);
int AveQuadProbes = ANOQPFL[0] + ANOQPFL[1] + ANOQPFL[2] + ANOQPFL[3] + ANOQPFL[4] +
    ANOQPFL[5] +
    ANOQPFL[6] + ANOQPFL[7] + ANOQPFL[8] + ANOQPFL[9];
int MaxQuadProbes = 0;
for(int b= 0 ; b < 10 ; b++){
    if( MaxQuadProbes < MNOQPFL[b])
        MaxQuadProbes = MNOQPFL[b];
}
System.out.println("Average Number of Quadratic Probes: " + AveQuadProbes);
System.out.println("Maximum Number of Quadratic Probes: " + MaxQuadProbes);
}
private static int nextPrime( int n )
{
    if( n % 2 == 0 )
        n++;

    for( ; !isPrime( n ); n += 2 )
        ;

    return n;
}

private static boolean isPrime( int n )
{
    if( n == 2 || n == 3 )
        return true;

    if( n == 1 || n % 2 == 0 )
        return false;

    for( int i = 3; i * i <= n; i += 2 )
        if( n % i == 0 )
            return false;

    return true;
}
}

```