```
public class LinearProbingHashTable
{
    private HashEntry [ ] array;
    private int currentSize = 0;
    public int TotalProbes = 0;
    public int MaxProbes = 0;
    public LinearProbingHashTable( int size )
    {
        allocateArray( size );
        makeEmpty( );
    }

    public float getLoad(){
        float load = (currentSize / array.length);
        return load;
    }
    public void insert( int x )
    {

        int currentPos = findPos( x );

        if( currentPos != -1){
            array[ currentPos ] = new HashEntry( x );
            currentSize++;
            return;
        }else{

            System.out.println("Unable to insert " + x + ".");
            return;
        }

    }

    private int findPos( int x )
    {
        int collisionNum = 0;
        int currentPos = hash( x, array.length );

        while( array[ currentPos ] != null && (array[ currentPos ].element != x) ){
            currentPos += 1;
            collisionNum++;
            if( currentPos >= array.length ) {
                currentPos -= array.length;
            }
        }
        if(collisionNum > array.length){
            return -1;
        }
        checkMax(collisionNum +1);
        TotalProbes += (collisionNum +1);
        return currentPos;
    }

    public void makeEmpty( )
    {
        currentSize = 0;
        for( int i = 0; i < array.length; i++ )
            array[ i ] = null;
    }

    public static int hash( int key, int TS )
    {
       key %= TS;
       if( key < 0 )
           key += TS;
      return key;
    }
    private void checkMax(int n){
        if(n > MaxProbes)
            MaxProbes = n;
        return;
    }
```

```
        private void allocateArray( int arraySize )
        {
            array = new HashEntry[ arraySize ];
        }


        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;
        }
}
```