```java
import java.util.Arrays;
import java.util.Scanner;
/*
* Nhan Vo
* SID#017771388
* CECS 328-lab#1
*/
public class Lab1 {
   public static void main(String[] args){
       Lab1 lab1=new Lab1();
       int input;
       int min=-1000;
       int max=1000;
       long totalRuns=100;
       long lTimeStore=0;
       long bTimeStore=0;
       long linearStart;
       long linearEnd;
       long binaryStart;
       long binaryEnd;

       Scanner scan= new Scanner(System.in);
       System.out.println("Please enter a positive number:");
       input=scan.nextInt();
       System.out.println("Your choosen number: "+input);
       int[] a=new int[input];
       for(int i=0;i<input;i++){
           a[i]=(int)(Math.random()*(max-min+1)+min);
       }


       //Part A
       System.out.println("--------------Part A--------------");
       Arrays.sort(a);
       for(int i=0;i<100;i++){
           int key=a[(int)(Math.random()*(a.length))];
           //System.out.println("New key is: "+key);

           linearStart=System.nanoTime();
           lab1.linearSearch(a,key);
           linearEnd=System.nanoTime();
           lTimeStore=lTimeStore+(linearEnd-linearStart);
           //System.out.println(lTimeStore);

           binaryStart=System.nanoTime();
           lab1.binarySearch(a,key);
           binaryEnd=System.nanoTime();
           bTimeStore=bTimeStore+(binaryEnd-binaryStart);
           //System.out.println(bTimeStore);
       }
       System.out.println("Average time to run 100 times linear search:
"+lTimeStore/totalRuns+" nano-second\n");

       System.out.println("Average time to run 100 times binary search:
"+bTimeStore/totalRuns+" nano-second\n");


       //Part B
       System.out.println("--------------Part B--------------");
       int key=5000;
       //System.out.println("New key is: "+key);
       //lTimeStore=0;
       linearStart=System.nanoTime();
       lab1.linearSearch(a,key);
       linearEnd=System.nanoTime();
       lTimeStore=linearEnd-linearStart;

       //bTimeStore=0;
       binaryStart=System.nanoTime();
       lab1.binarySearch(a,key);
       binaryEnd=System.nanoTime();
       bTimeStore=binaryEnd-binaryStart;

       System.out.println("Worst case time to run linear search: "+lTimeStore+"
nano-second\n");

       System.out.println("Worst case time to run binary search: "+bTimeStore+"
nano-second\n");

       /*
       Binary search runtime average is log(n) time. It take "bTimeStore"
nanosecond to finish, and the input size is 10^5
       Hence, the time to run one line of code will
equal=(bTimeStore/log(10^5))
       */
       long singleLineTime= (long)
(bTimeStore/(Math.log(Math.pow(10,5))/Math.log(2)));
       System.out.println("Runtime for a single line of code for binary search:
"+singleLineTime+" nano-second\n");

       /*
       With the input size=10^15, every single line of code take about 70 nano
seconds (from previous calculation),
       and worst runtime for binary search is log(n) and for linear search is
n.
       Then, the worse-case running time for:

       Binary search is: log(10^15)*(70*10^(-9)) = 3.4*(10^(-6)) second
```

```
        Linear search is: (10^15)*(70*10^(-9)) = 70000000 second
        */

        System.out.println("The worst case runtime for binary search: "+
(Math.log(Math.pow(10,15))/Math.log(2))*singleLineTime+" nano-second\n");

        System.out.println("The worst case runtime for binary search: "+
(Math.pow(10,15))*singleLineTime+" nano-second\n");




        //part C testing
        System.out.println("---------------Part C Testing---------------");
        double[] b={1,1.5,2,5,10,21};
        System.out.print("with array of: "+Arrays.toString(b)+" The answer is:
");
        System.out.println(lab1.modifiedbinarySearch(b)+"   Expected answer is
true\n");// should be true

        double[] c={1,5,12,17,19,27};
        System.out.print("with array of: "+Arrays.toString(c)+" The answer is:
");
        System.out.println(lab1.modifiedbinarySearch(c)+"   Expected answer is
false\n");// should be false

        double[] d={1,2,3,3.5,4,6,15,28};
        System.out.print("with array of: "+Arrays.toString(d)+" The answer is:
");
        System.out.println(lab1.modifiedbinarySearch(d)+"   Expected answer is
true\n");// should be true;

        double[] e={1,2,3,3.5,7,9,55,100,200};
        System.out.print("with array of: "+Arrays.toString(e)+" The answer is:
");
        System.out.println(lab1.modifiedbinarySearch(e)+"   Expected answer is
false\n");// should be false;
    }


    //Part C
    /*An algorithm with the running time of O(logn), similar to binary search
that
    checks if there is an i for which a[i]= i. */
    public boolean modifiedbinarySearch(double []a){
        int start=0;
        int mid=0;
    int end=a.length;
    while(start<end){
        mid=(end+start)/2;
        //System.out.println(mid);
        if(a[mid]==mid){
            return true;
        }
        else if(a[mid]>mid){
            end-=1;
        }
        else{
            start+=1;
        }
    }
    return false;
}



// linearSearch function
public boolean linearSearch(int[]a, int key){
    for(int i=0;i<a.length;i++){
        if(a[i]==key){
            return true;
        }
    }
    return false;
}


//binarySearch function
public boolean binarySearch(int[]a, int key){
    int start=0;
    int mid=0;
    int end=a.length;
    while(start<end){
        mid=(end+start)/2;
        if(a[mid]==key){
            return true;
        }
        else if(a[mid]>key){
            end=mid-1;
        }
        else{
            start=mid+1;
        }
    }
    return false;
}
```

```
}
```