Charlson So

1a.

O(n). algorithm will take the bigger the n value is

b. O(n+n)=O(n). Both for loops take as long as the value m.

c. O(n\*n+n\*10)=O(n^2) Nested for loop will take the longest. Therefore the notation is O(n^2).

d. O(n\*(n^1/2))=O(n^3/2) The nested for loops time complexity can be found by multiplying the size of the first for loop by the second for loop time complexity.

2a. Sequential Search is O(n) since finding the solution is based off the size of the search space.

b. for 56: 3\*0.5=1.5 ms

for 78: 4\*0.5=2 ms

for 100: 7\*0.5=3.5 ms

The sequential search will walk through the array until it finds the value. For value 56 and 78, the for loop will stop at their indexes; the number of times it loops is equal to its position. Therefore, the time it takes to find those integers would be the time it takes to loop multipled by the number of times looped. Since 100 is not in the array, the sequential search will check all the array elements with no answer.

c. Binary Search is a log­­2­(n)

d. 56: first for loop: checks middle, since middle>value, middle = last-1; the next for loop checks between index 0-2. Since middle is not value and new middle < value, middle=first+1. The next loop checks index 1-2. The new middle after this loops should be 1 since it is int type. Since middle<last, first = middle+1; the last for loops will only check index 2. Therefore, 4 loops.

78: 1 for loop. Since 78 is the middle value, the binary search will check the middle value and terminate.

100: Since 100 is not in the search space, it will loop the maximum number of times. I suspect the code in the lecture will loop infinitely number of times since there is no termination if the value is not found. Otherwise it should loop 4 times, since that is the maximum loops it takes to find an array.

3.

import java.util.\*;

public class fibo{

public static Boolean isPrime(int n){

Boolean t = true;

for(int i=2;i<n-1;i++){

if(n%i==0){

t=false;

}

}

return t;

}

public static void main(String[] args){

int fibn=20;

int [] fib = new int[fibn];

fib[0]=0;

if (fibn>0)

fib[1]=1;

for(int i =2; i<fibn;i++){

fib[i]=fib[i-1]+fib[i-2];

}

for (int j=0; j<fibn;j++){

System.out.println(Integer.toString(fib[j]));

}

//not checking numbers 0, 1, 1 ,2 since they are known primes and cannot be tested with this method.

for(int i=4;i<fibn;i++){

System.out.print("Fib number "+Integer.toString(fib[i])+" is ");

if (isPrime(fib[i])){

System.out.println(" a prime");

}

else{

System.out.println("not a prime");

}

}

}

}