/\*\*

\* In this code you will find a number of problems for a total of 50 points You

\* will need to identify

\* 1- three instances of O(n^2) notation 15 points

\* 2- when it is using a Linked List for a stack 10 points

\* 3- where it is using a sort and the type of sort it is 15 points

\* 4- Where a recursive method is broken and why it is broken 10 points

\*/

/\*\*

\* Name: Jesus Otteson

\*

\* Date: 06/05/2018

\*/

**public** **class** MidtermCode1CSIS2420 **extends** Comparable<E> {

**private** **double** lowTemp[] = **new** **double**[10];

**public** **void** weekLow(E[] t) {

**int**[] weektemp = **new** **int**[t.length];

**int** theLow;

**for** (**int** i = 0; i < weektemp.length; i++) {

theLow = i;

**for** (**int** k = i + 1; k < weektemp.length; k++) {

**if** (weektemp[k].compareTo(weektemp[theLow]) < 0) { This Loop is using

theLow = i; selection sort as there is its

} identifying a minimum here

**if** (theLow != i) { weektemp[k].compareTo(weektemp[theLow]) < 0

swap(weektemp, i, theLow);

}

}

}

}

**private** String cel2far(**double** temp) {

Double celsius = temp;

String tempString = celsius + " celsius is "

+ ((celsius \* 9 / 5.0) + 32) + " Fahrenheit";

**return** tempString;

}

**public** **int** newTemps(**int**[] temp) {

LinkList st = **new** LinkList();

**for** (**int** i = 0; i < temp.length - 1; i++) {

st.push(temp[i]);

}

**return** st.pop();

}

**public** **int** fixTemp(**int** f) {

**int** retTemp = 0;

**for** (**int** k = 0; k < f; k++) {

retTemp = f \* k;

}

**return** retTemp;

}

**int** tempChange(**int** number) {

**if** (number == 0) {

**return** 1;

}

tempChange\_i(number, 1);

}

**private** String far2cel(**double** temp) {

Double Fahrenheit = temp;

String tempString = Fahrenheit + " Fahrenheit is "

+ ((Fahrenheit - 32) \* (5 / 9.0)) + " celsius";

**return** tempString;

}

**int** tempChange\_i(**int** currentNumber, **int** sum) {

**if** (currentNumber == sum - 1) {

**return** sum;

} **else** {

**return** tempChange\_i(currentNumber + 1, sum \* currentNumber);

} This loop will never merge, it will continue to loop creating a data leak.

}

**public** **int** checktemp(**int** n) {

**int** sum = 0;

**for** (**int** j = 0; j < n; j++) {

**for** (**int** k = 0; k < n; k++) {

sum += lowTemp[j] \* lowTemp[k];

}

}

**return** sum;

}

**public** **double** adjustTemp(**double** t) {

**int** adjusted = 0;

**for** (**int** j = 0; j < t; j++) {

adjusted = fixTemp(j);

}

**return** adjusted;

}

}