/**********

Name: Noah Buchanan

Username: ua203

Problem Set: PS5

Due Date: July 7, 2020

**********/

Data Structures

July 6, 2020

1. I made this assuming by inserting a node you just meant at the end of the list and not in any sorted order

```
1: procedure InsertIntoCircular(node s)
2:     s.next = head;
3:     s.prev = tail;
4:     tail.next = s;
5:     head.prev = s;
6:     tail = s;
```

   O(1)

2. (a) find() : O(n)
   (b) insert() : O(n)
   (c) remove() : O(n)
   (d) printList() : O(n)
   (e) printListReversed() : O(n)

3. The time complexity starting at the head and being singly linked to insert at the end is O(n) time, just a loop to get to the end. You can improve this by using more memory to add a tail reference, so you can jump to the end and insert it without looping, this cuts down the run time to O(1). The reason the insert method listed in UALinkedList is O(n) with

a tail reference included is because it is inserting it in an arranged order descending.

4. time complexity with a head and tail reference of the delete function is O(n). There is no way to improve this because we cannot utilize arrays intuitive SizeOf(a)/SizeOf(type) method to jump straight to the index that we want, we must loop to the ith value in linked lists.

0.1 UACourse

```
public class UACourse {

        private int courseId;
        private String courseName;
        private String courseDescription;
        UACourse next;
        UACourse prev;
        private int accessCount = 0;

        public UACourse(int courseId, String courseName, String courseDescription) {
                this.courseId = courseId;
                this.courseName = courseName;
                this.courseDescription = courseDescription;
        }

        public String toString() {
                return getCourseId() + " " + getCourseName() + " " + getAccessCount();
        }


        public int getCourseId() {
                return courseId;
        }
        public void setCourseId(int courseId) {
                this.courseId = courseId;
        }
        public String getCourseName() {
                return courseName;
        }
        public void setCourseName(String courseName) {
                this.courseName = courseName;
        }
        public String getCourseDescription() {
                return courseDescription;
        }
```

```
        public void setCourseDescription(String courseDescription) {
                this.courseDescription = courseDescription;
        }
        public int getAccessCount() {
                return accessCount;
        }
        public void setAccessCount(int accessCount) {
                this.accessCount = accessCount;
        }


}
```

0.2 UALinkedList

```
public class UALinkedList {

        UACourse head;
        UACourse tail;

        public UACourse find(int courseId) {


                UACourse i = head;

                if (head.getCourseId() == courseId) {

                        i.setAccessCount(i.getAccessCount()+1);
                        return head;

                } else {

                        while (i.next != null && i.next.getCourseId() != courseId) {
                                i = i.next;
                        }
                        i.next.setAccessCount(i.next.getAccessCount()+1);
                        return i.next;
                }
        }

        public void insert(UACourse s) {

                if (head == null) {
```

```
                head = s;
                tail = s;
        } else {

                if (s.getCourseId() < head.getCourseId()) {

                        UACourse i = head;

                        while (i.next != null && s.getAccessCount() < i.next.getA
                                i = i.next;
                        }

                        if (i.next.getCourseId() == s.getCourseId()) {

                                System.out.println("ERROR - INVALID ID");
                        } else if (i.next == null) {

                                s.prev = i;
                                i.next = s;
                                tail = s;

                        } else {

                                s.prev = i;
                                s.next = i.next;
                                i.next.prev = s;
                                i.next = s;
                        }

                } else if (s.getCourseId() == head.getCourseId()) {

                        System.out.println("ERROR - INVALID ID");
                } else {
                        s.next = head;
                        head = s;
                        head.next.prev = head;
                }
        }
}

public void remove(UACourse c) {

        if (head != null) {
                if (head.getCourseId() == c.getCourseId()) {

                        head.next.prev = null;
```

```
                        head.next = head;
                } else {

                        UACourse i = head;

                        while (i.next != null && i.next.getCourseId() != c.getCo

                                i = i.next;
                        }

                        if(i.next == null) {
                                i.prev.next = null;
                                tail = i.prev;
                        } else {
                                i.prev.next = i.next;
                                i.next.prev = i.prev;
                        }

                }
        } else {
                System.out.println("List already empty");
        }
}

public void remove(int i) {

        if (head != null) {

                UACourse j = head;

                while (j.next != null && j.getCourseId() != i) {


                        j = j.next;
                }

                if(j.next == null) {
                        j.prev.next = null;
                        tail = j.prev;
                } else {
                        j.prev.next = j.next;
                        j.next.prev = j.prev;
                }

        } else {
                System.out.println("List already empty");
```

```
                }
        }

        public void printList() {

                UACourse i = head;

                while (i != null) {

                        System.out.println("|" + i.getCourseId() + "|" + i.getCourseName
                        i = i.next;
                }
        }

        public void printListReversed() {

                UACourse i = tail;


                while (i != null) {

                        System.out.println("|" + i.getCourseId() + "|" + i.getCourseName
                        i = i.prev;
                }
        }

}
```

0.3 class with main

```
public class RunTesterUALinkedList {

        public static void main(String[] args) throws IOException {

                UALinkedList list = new UALinkedList();

                BufferedReader br = new BufferedReader(new FileReader("records.txt"));

                String line = "";

                while((line = br.readLine()) != null) {
```

```
                String [] x = line.split(",");

                UACourse y = new UACourse(Integer.parseInt(x[2]),x[0],x[1]);

                list.insert(y);

            }

            list.printList();//a

            list.printListReversed();//b

            System.out.println(list.find(12345));//c

            System.out.println(list.find(45678));//d


            UACourse gpu = new UACourse(44332,"GPU Programming", "A fun course");//e
            list.insert(gpu);//e

            System.out.println(list.find(12345));//f

            list.remove(45678);//g

            System.out.println(list.find(55555));//h

            UACourse nlp = new UACourse(44332, "Natural Language Processing" , "NLP
            list.insert(nlp);//i

            System.out.println(list.find(44332));//j

            list.printList();//k
        }

}
```