```
Script started on 2021-02-09 21:45:44-0600
l_ladios@ares:~$ cat ScoreEntryLab.info
 /*****************************************************************************
     *                                                                           *
     *  NAME:  Leia Ladios                          CLASS:  CSC121-W01           *
     *                                                                           *
     *  Assignment: Lab P-3.37                      Level:   3                   *
     *                                                                           *
     *  Description:                                                             *
     *                                                                           *
        P-3.37: Maintains top ten scores of a game application. Implements
        add and remove methods(remove method is yet to come) of Section 3.1.1
                using a singly linked list instead of an array.

     *                                                                           *
    *****************************************************************************/

l_ladios@ares:~$ cat GameEntry.java
// modified implementation from Java book
public class GameEntry {

        private String name;
        private int score;

        public GameEntry(String name, int score) {
                this.name = name;
                this.score = score;
        }

        public String getName() {
                return name;
        }

        public int getScore() {
                return score;
        }

        public void setScore(int newScore) {
                score = newScore;
        }

        public int equals(GameEntry sec) {
                if(this.getName().equals(sec.getName()) && this.getScore()
                        == sec.getScore()) {
                        return 1;
                }
                else {
                        return 0;
                }
        }
        public String toString() {
                return "Name: " + name + "    Score: " + score;
        }
}
```

```
l_ladios@ares:~$ javac GameEntry.java
l_ladios@ares:~$ cat LinkesList.java
cat: LinkesList.java: No such file or directory
l_ladios@ares:~$ cat LinkedList.java
import java.lang.ArrayIndexOutOfBoundsException;
//Modified Java book's implementation of a Singly linked list.
public class LinkedList<T extends GameEntry> {

        //nested Node<T> class
        private static class Node<T> {
                //private instance variables
                private T data;
                private Node<T> next;


                //constructor Node class
                public Node(T data, Node<T> next) {
                        this.data = data;
                        this.next = next;
                }

                /*
                 * access method
                 * @returns data - data inside a Node.
                 */
                public T getData() {
                        return data;
                }

                /*
                 * access method
                 * @returns next - reference to the next Node
                 */
                public Node<T> getNext() {
                        return next;
                }

                /*
                 * setter method
                 * @param newNext - new next value being assigned
                 */
                public void setNext(Node<T> newNext) {
                        this.next = newNext;
                }
        }

        //private instance variables
        private Node<T> head;
        private Node<T> tail;
        private int size;

        //constructor
        public LinkedList() {
```

```
        head = null;
        tail = null;
        size = 0;
}

public T get(int index) throws ArrayIndexOutOfBoundsException {
    if(index < 0 || size <= index)
            throw new ArrayIndexOutOfBoundsException(index);

    Node<T> node = head;
    while(node.next != null && index > 0) {
        node = node.next;
        index--;
    }

     return node.getData();
}


/*
 * access method
 * @return size - size of linked list
 */
public int size() {
        return size;
}

/*
 * @return T/F - true if empty, false if filed.
 */
public boolean isEmpty() {
        return size == 0;
}

/*
 * @return first Element of list
 */
public T first() {
        return head.getData();
}

/*
 * @return last Element of list
 */
public T last() {
        return tail.getData();
}

/*
 * adds to the top of the list
 * @param data - data for Node to be inserted.
 */
public void addFirst(T data) {
        head = new Node<T>(data, head);
        if(size == 0) {
                tail = head;
        }

        size ++;
}

/*
 * adds to the end of the list
 * @param data - data for Node to be inserted.
 */
public void addLast(T data) {
        Node<T> newest = new Node<T>(data, null);
        if(isEmpty()) {
                head = newest;
        }
        else {
                tail.setNext(newest);
        }
        tail = newest;
        size++;
}

/*
 * removes the first Node on the list.
 * @return ret - data of Node being removed.
 */
public T removeFirst() {
        if(isEmpty()) {
                return null;
        }
        T ret = head.getData();
        head = head.getNext();
        size--;
        if(size == 0) {
                tail = null;
        }
        return ret;

}

/*
 * add score into list only if it is higher than a score already on
 * the board.
 */
public void addEntry(T entry) {

        if(size < 2) {
                if(isEmpty() == true || entry.getScore() > head.getData().g
                        addLast(entry);
                }
                else if(isEmpty() == false) {
                        Node<T> temp = head;
```

```
                        head = new Node<T>(entry, temp);
                }
        }
        else if(size <= 10) {
                organize(entry);
        }

}

private void organize(T entry) {

        Node<T> scoreEntry = new Node<T>(entry, null);
        boolean added = false;
        Node<T> current = head;

        while(current.getNext() != null && !added) {
                if(current.getData().getScore() <= scoreEntry.getData().get
                  && current.getNext().getData().getScore() >= scoreEntry.g
                        Node<T> temp = current.getNext();
                        scoreEntry.setNext(temp);
                        current.setNext(scoreEntry);
                        size++;
                        added = true;
                }
                else if(scoreEntry.getData().getScore() < head.getData().ge
                        addFirst(scoreEntry.getData());
                        added = true;
                }
                else {
                        added = false;
                        current = current.getNext();
                }
        }

        if(added != true) {
                addLast(entry);
        }

        if(size > 10) {
                removeFirst();
        }

}

public void remove(int index) {

        int i = 0;
        Node<T> current = head;
        Node<T> prev = null;
        boolean removed = false;

        if(isEmpty()) {
                System.out.println("There is nothing to remove");
        }
                if(index == 1) {
                        removeFirst();
                }
                else {
                        while(current.getNext() != null && !removed) {
                                if(i == (index - 1)) {
                                        prev.setNext(current.getNext());
                                        size--;
                                        removed = true;
                                }
                                else{
                                        prev = current;
                                        current = current.getNext();
                                        i++;
                                }
                        }

                        if(index == size) {
                                prev.setNext(null);
                                size--;
                        }
                }
        }

        public String toString() {
                int i = 0;
                String build = "";
                Node<T> current = head;
                while(i < size && current.getNext() != null) {
                        build += (i+1) + ") " + current.getData().toString() + "\n
                        current = current.getNext();
                        i++;
                }
                build += (i+1) + ") " + tail.getData().toString();
                return build;
        }
}

l_ladios@ares:~$ javac LinkedList.java
l_ladios@ares:~$ cat PlayerListMain.java
import java.util.Scanner;
public class PlayerListMain {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Scanner in = new Scanner(System.in);

                LinkedList<GameEntry> scoresList = new LinkedList<GameEntry>();


                scoresList.addEntry(new GameEntry("Carl", 78));
                scoresList.addEntry(new GameEntry("James", 80));
                scoresList.addEntry(new GameEntry("Marjorie", 92));
                scoresList.addEntry(new GameEntry("Mary", 90));
```

```
scoresList.addEntry(new GameEntry("Kami", 121));
scoresList.addEntry(new GameEntry("Clarie", 150));
scoresList.addEntry(new GameEntry("Maybell", 139));
scoresList.addEntry(new GameEntry("Timmy", 40));



boolean goAgain = true;
while(goAgain) {
        boolean done = false;
        System.out.println("\nWould you like to\n 1)add \n 2)remove
                        "\t(Please enter 1 or 2)");
        int choice = in.nextInt();

        while(!done) {
                if(choice == 1) {
                        System.out.println("\nPlease enter a score

                        System.out.print("Enter Score: ");
                        int score = in.nextInt();
                        System.out.print("Enter Name: ");
                        String name = in.next();
                        scoresList.addEntry(new GameEntry(name, sc

                        System.out.println("\nWould you like to ad
                        String keepGoing= in.next();

                        if(keepGoing.equalsIgnoreCase("N")) {
                                done = true;
                        }
                        else if(keepGoing.equalsIgnoreCase("Y")){
                                done = false;
                        }

                }
                else if(choice == 2) {

                        System.out.println("\n" + scoresList);
                        System.out.println("\nWhich entry would yo

                        scoresList.remove(in.nextInt());

                        System.out.println("\nWould you like to re
                        String keepGoing= in.next();

                        if(keepGoing.equalsIgnoreCase("N")) {
                                done = true;
                        }
                        else if(keepGoing.equalsIgnoreCase("Y")){
                                done = false;
                        }

                }
                        }
                        System.out.println("\nDo you want to \n1)remove/add another
                                        "\t(Please enter 1 or 2)");

                        int toGoAgain = in.nextInt();

                        if(toGoAgain == 2) {
                                goAgain = false;
                        }
                }

                System.out.println(scoresList);

        }

}
```

```
l_ladios@ares:~$ javac PlayerListMain.java
l_ladios@ares:~$ java PlayerListMain

Would you like to
 1)add
 2)remove a score?      (Please enter 1 or 2)
1

Please enter a score
Enter Score: 300
Enter Name: Leia

Would you like to add another score? (Y/N)
Y

Please enter a score
Enter Score: 250
Enter Name: Kim

Would you like to add another score? (Y/N)
Y

Please enter a score
Enter Score: 160
Enter Name: Jeremy

Would you like to add another score? (Y/N)
Y

Please enter a score
Enter Score: 85
Enter Name: Kain

Would you like to add another score? (Y/N)
N
```

```
Do you want to
1)remove/add another score
2)exit ?        (Please enter 1 or 2)
1

Would you like to
 1)add
 2)remove a score?      (Please enter 1 or 2)
2

1) Name: James    Score: 80

2) Name: Kain    Score: 85

3) Name: Mary    Score: 90

4) Name: Marjorie    Score: 92

5) Name: Kami    Score: 121

6) Name: Maybell    Score: 139

7) Name: Clarie    Score: 150

8) Name: Jeremy    Score: 160

9) Name: Kim    Score: 250

10) Name: Leia    Score: 300

Which entry would you like to remove? (Select a number)
3

Would you like to remove another score? (Y/N)
N

Do you want to
1)remove/add another score
2)exit ?        (Please enter 1 or 2)
2
1) Name: James    Score: 80

2) Name: Kain    Score: 85

3) Name: Marjorie    Score: 92

4) Name: Kami    Score: 121

5) Name: Maybell    Score: 139

6) Name: Clarie    Score: 150

7) Name: Jeremy    Score: 160

8) Name: Kim    Score: 250

9) Name: Leia    Score: 300
l_ladios@ares:~$ exit
exit

Script done on 2021-02-09 21:48:08-0600
```