```
// Cooper Kelley (clk200002)

// pseudocode for project 3: updated ticket reservation systems

// objectives of this lab:
// 1) create and manipulate a linked list of linked lists
// 2) utilize classes to create a basic data structure

// auditorium class
generic auditorium class {
    // members:
    generic node first      // acts as head pointer

    // default constructor
    auditorium() {
        first = null;
    }

    // mutator
    void setFirst(generic type newFirst) { first = newFirst; }
    // accessor:
    generic node getFirst() { return first };
}

// seat class
seat class {
    // members:
    int row;
    char seat;
```

```
        char tType;

        // default constructor
        seat() {
                row = 0;
                seat = '';
                tType = '';
        }

        // overloaded constuctor
        seat(int r, char s, char tT) {
                row = r;
                seat = s;
                tType = tT;
        }

        // mutators
        void setRow(int r) {row  = s;}
        void setSeat(char s) {seat = s;}
        void setType(char tT) {tType = tT;}
        // accessors
        int getRow() {return row;}
        char getSeat() {retrun seat;}
        char getType() {return type;}
}

// node class
generic class node {
```

```
// members:
generic node next;
generic node down;
generic node prev;
generic payload;

// default constructor
node() {
        next = null;
        down = null;
        prev = null;
        payload = null;
}

// overloaded constuctor
node(node n, node d, node pr, generic pl) {
        next = n;
        down = d;
        prev = p;
        payload = pl;
}

// mutators
void setNext(node n) {next = n;}
void setDown(node d) {down = d;}
void setPrev(node pr) {prev = pr;}
void setPayload(generic pl) {payload = pl;}
// accessors
generic node getNext() {return next;}
```

```
        generic node getDown() {return down;}

        generic node getPrev() {return prev;}

        generic payload() {return payload;}

}


// input validation


// how to implement the best available seats


// a list of at least 10 test cases you will check during testing

//                  specific input is not necessary

//                  describe what you are testing


// main:

main () {

        create an auditorium object

        read file into object

        determine if seats are available:

                reserve if available

                determine if number of seats requested is available in a row

                        if not, calculate the closest seat using eucledian distance

        wrtie the report to console

        write the auditorium object to an output file


        }


}
```