Project Three Pseudocode

```
Main_Function {

    Create an Auditorium object lets call it aud_obj.

  • Loop (do…while not exit)
      o Print main menu
      o Get user input
      o If user does not want to exit
          ▪ Display auditorium (calling a method inside Auditorium class)
          ▪ Ask user for row, seat, and quantities (adult, child, senior)
          ▪ Validate input for each input
              • Loop if not valid
          ▪ Check if seats are available by calling appropriate method inside the Auditorium
            class
              • If available reserve seats
          ▪ If seats not available
          ▪ Find best available seats by calling appropriate method inside the Auditorium
            class
              • Ask user to reserve best available (if found)
              • If user accepts
                  o Reserve seats
    Print report by calling appropriate method inside the Auditorium class
    Main is  similar to Project-1, but most of the array functions will be missing as they are taken
    care in Auditorium class.
    }
```

Auditorium Class {

Have a object called **first** of type Node to store the first node in the Linked List of Linked List.
Have appropriate Constructors.
One of the constructors will call the set_Auditorium routine

```
  private void set_Auditorium(String filename){
    Both the constructor and resetting method will call this method.
```

```
This method will set or reset the Auditorium from a file.
 first = null;

Open the file with the given name filename
For every line in the file, do the following{
   For every line, every single character indicates a Seat.
Create a seat object per character and then create node with the Seat object as payload. Make sure the Nodes become a Linked List of Linked List.


      }// end of reading the next line
 }
```

```
  public void print_Auditorium() {

 This method will print the Auditorium , in the proper style

    Node<Seat> current = first
        if(current==null) {
            print("Auditorium Object Empty") This should never happen but better to not
            ignore NullPointerException
            return
        }

    Node<Seat> current_row = first this pointer will always point to the first Node of the current
row
    int row_number= 0, to keep track of the row_number being printed
    print("\n\nAuditorium Seating")
        First Lets print the Characters to indicate Seat number
        for( i=65;i<65+no_of_seats;i++) {
            print(  Character with Ascii Number i )
        }

        while(current_row !=null) {
            row_number++
          print the proper row number with proper spacing
            while(current != null)
            {
                Seat s = current.get_payload()
                if(s.get_ticket_type() != '.')
                    print('#')
```

```
                        else
                                print(s.get_ticket_type())

                        we have printed the current seat lets go to the next


                        current =  current.get_next();



                }current ! = null
            Row printing is done, print a new line for the next row

                current_row =  current_row.get_down()
                current = current_row
            }current_row != null

            All Nodes have been printed into Lines.


} end of print_Auditorium
```

```
  public boolean check_Auditorium(int tt, int row, int seat) {
```
This method will check if the required seats are free.

```
        if(first == null)
                return false;

        Node<Seat> current = first;



        Lets go to the appropriate row and seat by jumping using down and next pointer.
        int i=1;
        while(i<row) {
            current= current.get_down() Getting to the correct row
            i++;

        }

            i =65;  Remember the seat contains the Ascii number of the character
        while(i<seat) {
                current= current.get_next();// Getting to the correct seat
                i++

        }
        i=1;
        while(i<=tt) {
```

Checking if the seats are available for booking

```
if(current.get_payload().get_ticket_type() !='.')

                        return false
                else {
                                current= current.get_next()
                                i+
                }
        }
        return true;
  } end of check Auditorium
```

```
    public void book_Auditorium(int row, int seat, int a, int c, int s ) {
```

This method will allocate the tickets in the proper location

First lets get to the correct row and seat by using down and next pointer. This is done very similar to Check_Auditorium method.

Once the correct starting row and starting seat is reached booking begins.

```
        Node<Seat> current = first
```

Lets book the correct adult seats first

```
        i=1 , a pointer to keep track of number of seats assigned A
        while(i<=a) {
                        current.get_payload().set_ticket_type('A')
                        current = current.get_next()
                        i++
                        }
```

Let’s book the correct number of child seats

```
        i=1
        while(i<=c) {
                        current.get_payload().set_ticket_type('C')
                        current = current.get_next()
                        i++
                        }
```

Let’s book the correct number of senior seats

```
        i=1;
        while(i<=s) {
                        current.get_payload().set_ticket_type('S')
                        current = current.get_next()
                        i++
            }
        }// end of book auditorium
```

`Public boolean best_Available(int total_seats_required) {`

- This routine will compute the best available seats in the entire auditorium.
- The best available seats are the ones closest to the center.
- To compute the best seats, we will scan the auditorium row by row and get the best seats in each row.
- It will help to store the center of the auditorium as global floating-point values.
- It will also help to have a distance calculating routine that will calculate the distance between the given point and the center of the auditorium.
- Another helper routine will be required to figure out if the seats are available that is if required number of seats are available together in a single row.
- Once we get a particular set of seats, we will calculate the center of seats with center of the auditorium and store it in an array.
- If there are multiple choice in a single row, one that closest to the center will be stored. if there is a tie, selection with lower seat value will be used.

- Two different arrays to store distance and selection could be used.
- Once we get the array where in each element in the array is the best seat in a particular row. We will loop through the array to find the selection closest to the center of the auditorium. If there is a tie row that is closest to the center will be used, if there is a tie in that we will choose the lower row value.
- If we could find alternate seats, we will return true, if we found nothing we will return false.
- It is a good idea to store best available starting seats as global parameters and have other routine decide whether to book the seats or not based on user option.

} End of Best Available

`public void write_File()` {

Open a new file to write called “A1.txt”.

```
  Node<Seat> current = first
  Node<Seat> current_row = first , this pointer will always point to the first Node of the current row.
          while(current_row !=null) {
               while(current != null)
               {
                    Seat s = current.get_payload()
                    Print the ticket type into the file
                    current =  current.get_next()
                         }current ! = null
                    Row printing is done, print a new line for the next row.

                    current_row =  current_row.get_down(), command to go to the next row.
                    current = current_row
                }current_row != null
                Close the file properly.


  } end of write File
```

```
public void calculate_Output(){

          current = first;
   current_row = first; // this pointer will always point to the first Node of the current row
         Initialize the following variable  sold_adult =0, sold_child = 0 and sold_senior = 0.

                while(current_row !=null) {
                    while(current != null)
                    {
                        Seat s = current.get_payload();
                            if(seat s 's ticket type is equal to 'A')
                                sold_adult++;
                             else if(seat s 's ticket type is equal to 'C')
                            sold_child++;
                         else if(seat s 's ticket type is equal to 'S')
                            sold_senior++;
                        current =  current.get_next();
                    } current != null
            One row is done.
                    current_row =  current_row.get_down();
                    current = current_row;
                }current_row != null

     Message is printed appropriately, Total sales price is calculated as follows
Total Sales = sold_adult * 10 + sold_child * 5 + sold_senior * 7.5
     Total Sales is printed appropriately.

} end of calculate_Output
} end of the class Auditorium
```

Node Class and **Seat Class** are straight forward with private data members and public accessor and mutators.