CS 1337 – PROJECT 3 – Avengers 4 Ticket Reservation System (Extended Edition)

Pseudocode Due: Section 001 (M/W) – 10/15 by 10:00 PM
Section 002 (Tu/Th) – 10/16 by 10:00 PM

Project Due: Section 001 (M/W) – 10/31 by 10:00 PM
Section 002 (Tu/Th) – 11/1 by 10:00 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- All project deliverables are to be submitted in eLearning.
- Zip all of the source files into a single zipped file
    - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
    - Please review the submission testing information in eLearning on the Course Homepage
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile using gcc 7.3.0 or higher with the following flags enabled
    - -Wall
    - -Wextra
    - -Wuninitialized
    - -pedantic-errors
    - -Wconversion
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.** (-5 points)

Objectives:

- Implement a structure that includes pointer and non-pointer members
- Utilize pointers to dynamically allocate memory for structures
- Create and manipulate a multi-directional linked list in C++.

Problem: In preparation for the release of Avengers 4, you have been hired by the owner of a small movie theater to develop the backend for an online ticket reservation system. Patrons will be able to reserve seats in one of three auditoriums. Once the patron has selected an auditorium, the program should display the current seating arrangement and allow the patron to select seats. A report should be generated at the end of the program to specify for each individual auditorium and overall for all auditoriums how many seats were sold/unsold and how much money was earned.

Pseudocode: Your pseudocode should describe the following items

- Main.cpp
    - List functions you plan to create
        - Determine the parameters

        - Determine the return type
        - Detail the step-by-step logic that the function will perform
    - Detail the step-by-step logic of the main function

Structure

- Node
    - Members
        - Row (integer)
        - Seat (character)
        - Reserved (boolean)
        - Ticket type (character)
        - Up (Node pointer)
        - Down (Node pointer)
        - Left (Node pointer)
        - Right (Node pointer)

Details

- The seating arrangement for each auditorium will be stored in a file named *A1.txt*
- Each line in the file will represent a row in the auditorium. The number of rows in the auditorium is unknown to you.
    - There will be a newline character after each line in the file except for the last line which may or may not have a newline character.
- The number of seats in each row of the auditorium will be the same.
- No row will have more than 26 seats.
- The auditorium will be held in memory by a linked set of structures connected by pointers that form a grid (-15 points if not)
    - You should do this as you read the file
    - Store the row and seat in each node that you create
    - Also mark if the seat is reserved and if so what type of ticket was bought
    - Link the node to its neighbors in the grid.
- Empty seats are represented by a period (.).
- Reserved seats are represented by a letter (A, C or S) in the file
    - This will be used to create reports
    - A =adult
    - C = child
    - S = senior
- Reserved seats will be represented by a hashtag (#) on the screen
    - The user does not need to know what type of ticket was sold, just that a seat is reserved.
- There is no need to worry about multiple screenings or reserving seats on a specific day.
- Ticket prices are as follows:
    - Adult - $10
    - Child - $5
    - Senior - $7.50

User Interface and Input: Present a user-friendly menu system for the user to select the auditorium. First ask for the auditorium:

```
1. Reserve Seats
2. Exit
```

Loop the menu until the user decides to quit. Imagine this is for a ticket kiosk in the lobby of the theater.

If the user wants to reserve seats, display the current seating availability for that auditorium. An example seating chart is provided below for an auditorium with 5 rows and 20 seats per row.

```
  ABCDEFGHIJKLMNOPQRST
1 ...##..#####........
2 ########....####..##
3 .........##.........
4 #.#.#.#.#.#.#.#.#.#.
5 ########.#####.#####
```

The rows are numbered and the seats are identified in each row by a letter of the alphabet

After the seating chart has been displayed, prompt the user for the following information in the order below:

- Row number
- Starting seat letter
- Number of adult tickets
- Number of child tickets
- Number of senior tickets

Assume that the user wants to reserve sequential seats to the right of the first seat entered. Adult tickets will be reserved first, followed by child and then senior. All seats must be open for a reservation to be processed.

If the desired seats are not available, offer the user the best available seats that meet their criteria **in the entire auditorium**. The best available seats are the seats closest to the middle of the auditorium.

- Think of the distance between 2 points
- Use the distance between the center of the selection and the center of the auditorium.
- In the event of a tie for distance, the row closest to the middle of the auditorium should be selected.
- In the event of a tie for closest row, use the row with the smaller number

State to the user what the best available seats are and then ask if they would like those seats. Prompt the user to enter a **Y** to reserve the best available or **N** to refuse the best available. If the user declines the best available seats, return the user to the main menu. If the user accepts the best available seats, reserve them and display a confirmation to the screen. Once the selection has been processed, return to the main menu**.** If there are no alternate seats available, display an appropriate message to the user instead of a prompt and return to the main menu.

When prompting the user for input, expect anything. Do not assume any input will be valid. If you ask for a number, the user could put in a floating point number, symbols or maybe even the Gettysburg Address (or at least

a sentence or two). Make sure that the user selection does not go out of bounds of the auditorium. If invalid input is entered, loop until valid input is received.

User Interface Workflow: Please do not add extra prompts since this will cause a mismatch in the input which will either force the program to throw an exception or cause the program to perform an unintended operation.

- Display main menu
- Prompt for input
- If user is reserving tickets
    - Prompt for row
    - Validate – loop until valid
        - Valid row = row number listed in auditorium display
    - Prompt for starting seat
    - Validate – loop until valid
        - Valid seat = seat number listed in auditorium display
    - Prompt for number of adult tickets
    - Validate – loop until valid
        - Valid ticket number = number >= 0
    - Prompt for number of child tickets
    - Validate – loop until valid
        - Valid ticket number = number >= 0
    - Prompt for number of senior tickets
    - Validate – loop until valid
        - Valid ticket number = number >= 0
    - If seats unavailable
        - Display best available
        - Prompt user to reserve (Y/N)
    - If reserved, confirm reservation
    - Return to main menu
- Loop to top of workflow until user selects exit

Output: At the end of the program, write the current status of the auditorium back to the file. Remember to write the auditorium reservations using A, C and S to identify the type of ticket sold. Also, display a formatted report to the console. Make sure each column lines up properly (no jagged columns). Include the following information in the order given:

- Total Seats in Auditorium
- Total Tickets Sold
- Adult Tickets Sold
- Child Tickets Sold
- Senior Tickets Sold
- Total Ticket Sales – total amount of money collected for tickets in the auditorium

All values except total ticket sales will be an integer value. Total ticket sales will be a floating-point value rounded to 2 decimal places and formatted with a dollar sign before the first digit of the number.