

## ENGG1111 Computer Programming and Applications - Assignment 2

### Second Semester, 2015-2016

#### The Problem

In this assignment, we are going to produce a program which provides daily operations for GiL cinema.

#### GiL cinema

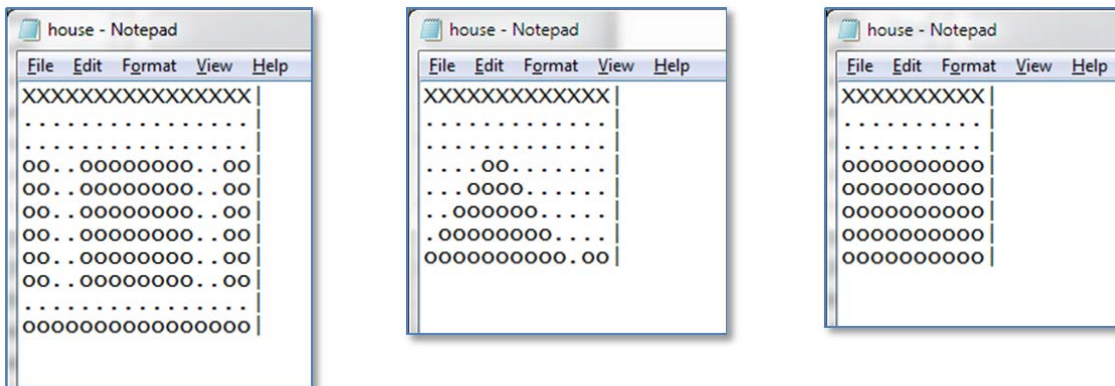
This is a small cinema with only 1 house in a maximum 20x20 seating arrangement. That is, there are maximum 20 seats in a row, and maximum 20 rows in the house. It operates daily from morning to night, showing a maximum of 10 movies each day. Like other cinemas, customers can buy ticket for movies showing today, tomorrow, and the day after tomorrow.

#### Daily operations

Sam operates the ticket counter, his daily routine is listed below:

- Open counter and get ready to sell tickets.
- Sell tickets according to customer selection.
- At the end of the day, generate ticket selling statistics for his boss.

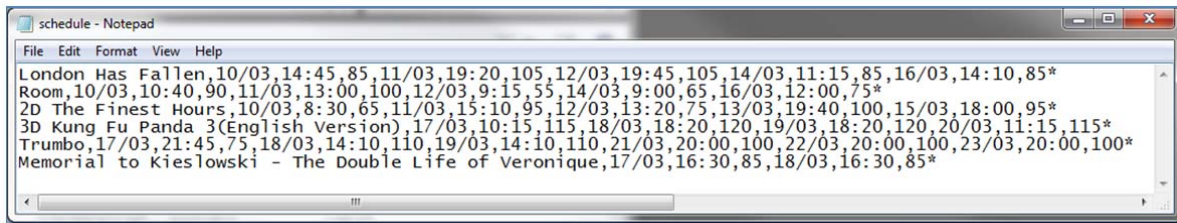
In order to produce a smooth daily operation, Sam works with these data files:



Input filename: **house.txt**

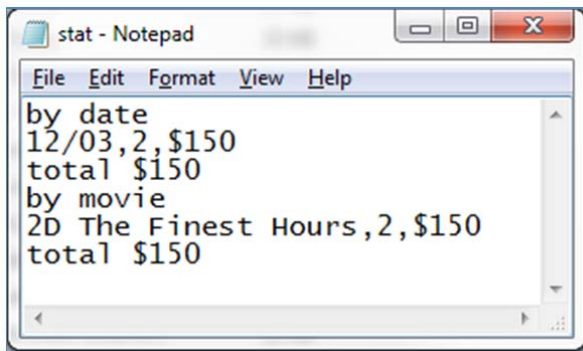
This file shows the seating plan of the cinema house. The first line represents the screen, the next 2 lines represents the space between the screen and the first row. A dot ('.') represents a space, 'o' represents a seat. Each line is terminated by '|', and there is no empty line in this file.

The above displays three sample seating plans. Any seating plan having a size not exceeding 20 rows × 20 columns is considered valid.

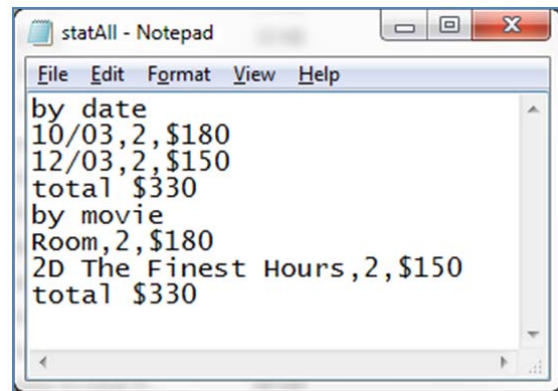


Input filename: **schedule.txt**

This file has one or more lines. Each line gives all schedules for one movie title: show date, start time, ticket price. If the movie is showing more than once, repeating sets of show date, show time, ticket price will be shown on the same line. Every piece of information is separated by comma. \* denotes the end of the line. In the above sample schedule, the movie "3D Kung Fu Panda 3(English Version)" is showing many times in 4 days, while "Room" only shows once.



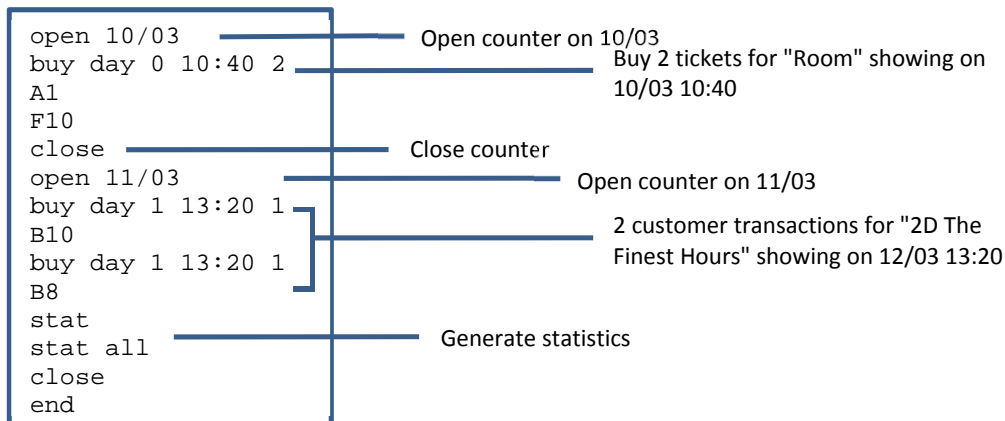
Output filename: **stat.txt**



Output filename: **statAll.txt**

Both files are output files listing number of tickets sold by movie and by date. If there is no ticket sold in a date or for a movie, the output is omitted. Note that total by date and total by movie should have the same value.

According to the sample schedule in previous page, the above statistics are generated by these commands:



What kind of ticketing information to record? You are going to design this file.

### Design a data file to store ticketing information

This system models the daily ticketing of this cinema, therefore tickets bought by customers today cannot be made available the next day. Sam needs your help to design this file. At the end of the day, before Sam closes the system, all useful information needs to be saved so that the system will continue to operate the next day. In other words, if a customer bought D5 D6 for "3D Kung Fu Panda 3(English Version)" showing on 17/03 10:15, those 2 tickets cannot be made available to another customer the next day.

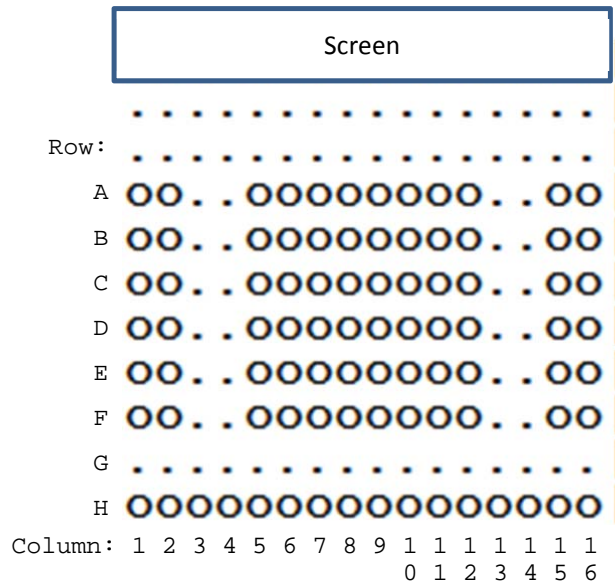
### Your task

Develop a program which provides the following operations for Sam to manage the ticketing counter.

command	What to do
<b>open</b> dd/mm	This is the beginning of a new day (dd/mm). The system opens necessary file(s), updates memory, and gets the ticket counter ready to sell tickets. dd/mm is entered by the operator.  This command will always be the first command to run.
<b>close</b>	This is the end of a day. The system updates necessary file(s) and closes the ticket counter.  If this command is not executed, no update will be made to the file(s).
<b>show all</b>	Display all movie schedules. This is equivalent to typing these commands: show day 0 show day 1 show day 2
<b>show day</b> x	Display all movie schedules for day x (x=0 means today, x=1 means tomorrow, x=2 means the day after tomorrow)
<b>show day</b> x hh:mm	Display seating plan for a specific day (x=0, 1, or 2) and start time (hh:mm).
<b>buy day</b> x hh:mm N	Buy N tickets for a specific date (x=0, 1, or 2) and start time (hh:mm).  This command will be followed by N lines of valid seats. If the seat entered is not valid or available, that input will be ignored.
<b>stat</b>	Calculate statistics for today and write to "stat.txt". This command will be performed before "close", and will only consider those transactions from "open" to this moment.
<b>stat all</b>	Calculate accumulated statistics up to this moment and write to "statAll.txt". This command can be called before or after "close".
<b>end</b>	End the program. No need to do any update of file(s).

## Program testing

The following shows a sample seating plan. Row numbering starts from A (closest to screen), and column numbering starts from 1 (left to right).



Sample executable and data files are available in Moodle for testing purpose. Please download from Moodle according to your Operating System. Do not attempt to reverse engineer the executable file.

## Working with data files

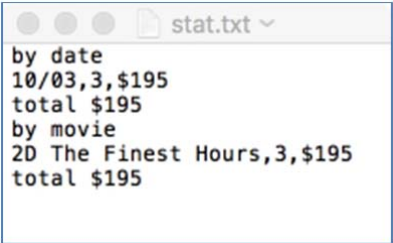
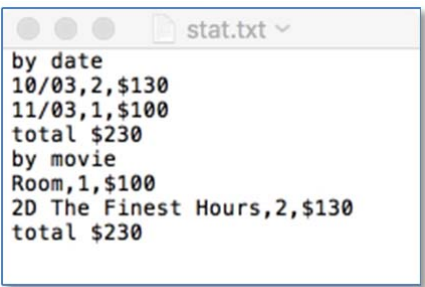
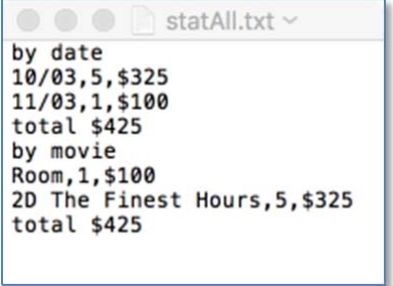
If you are using Windows OS,

- Please use ONLY the data file for Windows provided in Moodle.
- Put all the data files in the same folder as your cpp file.

If you are using Mac OS,

- Please use ONLY the data file for Mac provided in Moodle.
- Put all the data files in your home folder.

The following shows two sample runs using data from sampleHouseA.txt and sampleScheduleA.txt available in Moodle. Some screen output is too long and therefore omitted here. Please use the executable file in Moodle to generate more samples.

Sample input and <b>output</b> (some screen outputs are ignored)	Output file - stat.txt
<pre> open 10/03 show day 0 8:30 ..... ... SCREEN ... ..... 0000000001111111 1234567890123456  A oo 00000000 oo B oo 00000000 oo C oo 00000000 oo D oo 00000000 oo E oo 00000000 oo F oo 00000000 oo G H 0000000000000000 buy day 0 8:30 3 A10 F12 E14 E15 stat close end </pre>	 <pre> by date 10/03,3,\$195 total \$195 by movie 2D The Finest Hours,3,\$195 total \$195 </pre>
Sample input and <b>output</b> (some screen outputs are ignored) this sample continues from above	Output files - stat.txt and statAll.txt
<pre> open 10/03 show day 0 8:30 ..... ... SCREEN ... ..... 0000000001111111 1234567890123456  A oo 00000-oo oo B oo 00000000 oo C oo 00000000 oo D oo 00000000 oo E oo 00000000 -o F oo 0000000- oo G H 0000000000000000 buy day 0 8:30 2 E15 B10 B1 buy day 1 13:00 1 H8 stat stat all close end </pre>	 <pre> by date 10/03,2,\$130 11/03,1,\$100 total \$230 by movie Room,1,\$100 2D The Finest Hours,2,\$130 total \$230 </pre>  <pre> by date 10/03,5,\$325 11/03,1,\$100 total \$425 by movie Room,1,\$100 2D The Finest Hours,5,\$325 total \$425 </pre>

## Assumptions you can / cannot made in the program

You can base on the following assumptions when writing the program:

- All input data files (house.txt, schedule.txt) exist. They are non-empty and contain at least one line.
- There will be no empty line at the end of the input files.
- Maximum 50 movie titles in schedule.txt.
- Maximum 400 seats in the house (max 20 rows, max 20 columns).
- Maximum 10 movie shows per day.
- Data files given are samples only. The data files used for marking have the same format but different contents.
- All user input will be in correct format.
- User input date and time will always be valid, that means the input date and time will match with one and only one movie schedule.
- All dates are assumed in the year 2016, with the format dd/mm. e.g. 21/03, 01/01
- User input seat selection (e.g. D9) contains a row (A-T) and a column (1-20), but this seat might not be available for that movie.
- Appropriate size of array must be created. Do not create oversized array.
- Do not assume any default initialization of variables and array. Write your own initialization codes.
- No need to generate error message. If your program considers the input is wrong, just ignore that input.
- Screen output will not be marked.
- Output files (stat.txt and statAll.txt) will be marked by computer, formatting must be observed strictly.

## Rules and regulations

[I/O formats] Your program will be marked by machines. Therefore, the given input (from keyboard, from file) and output (to file) formats should be followed strictly. All cout will not be marked by computer. Each test case will be marked as "correct" or "incorrect", no partial marks will be given to a test case.

[Programming skills] You must demonstrate the use of struct, arrays, functions, strings, and file in your program. Other programming skills outside the scope (lecture 1-8) cannot be used. Improper logic, such as using "break" to force a loop to terminate, will cause mark deduction. If you are not sure, please ask.

[Plagiarism] Plagiarism is strictly prohibited! Zero or negative score will be given to both the source and copy if discovered. You must not copy or let others copy your work. It's your own responsibility to protect your files and prevent others from copying your program directly or indirectly (e.g. obtain your program through another person). The following constitutes an act of plagiarism:

- submitting another student's work as your own, or outsourcing;
- direct copying (full or partial);
- studying someone else's program and then rewriting it as your own;
- group work (i.e. working side by side, and discuss the program line by line);
- providing your assignment as the source for any of the above actions.

**Submission (Late submission will not be marked)**

Only limited test cases are made public for testing. You are encouraged to create other test cases for testing purpose. Your program will be tested against a number of cases, and each case will be worth a certain number of marks.

**Submission of your program (*yourUNo\_a2.cpp*) by 6:00pm 25 Apr (Mon) via Moodle**

**Put your name and your 10-digit UNo as comment in the first line of your cpp program.** Submit a single **.cpp file** (i.e. *yourUNo\_a2.cpp*). Unless Moodle is proved to be out of service within 30 minutes before the deadline, the deadline will not be extended under all circumstances. Submitting a wrong file is not an excuse for late submission. Make sure you have double checked before submission.

**Submission of your paper report during lecture on 28 Apr (Thu)**

Prepare the followings (not more than 2 pages of one A4 paper):

- Your name, university no, email.
- Screen captures showing: (1) all function declarations (the first line of every function), (2) user defined structs, (3) major arrays declared in main and global.
- How does your program record day-to-day selling of tickets? Explain the design and use of your user defined file.
- [Bonus] Share 1 smart feature of your program **OR** write a smart add-on command to get the bonus.

**Marking scheme [total mark: 100]**

- 40 marks for program testing
- 20 marks for design and use of the intermediate data file. More marks will be given for a small and effective design.
- 20 marks for design and use of variables. More marks will be given for:
  - Local instead of global
  - Meaningful use of struct and array
  - Proper size of array
- 10 marks for design and use of functions. More marks will be given for:
  - Passing array to function instead of using global array
  - Meaningful functions to break down the logic
- 10 marks for the report.
- 10 marks for the bonus.
- Deduction (max 20 marks) for incorrect input/output format, wrong filename.
- Deduction (max 50 marks) for the use of break or skills outside our scope.