# CS 302 Data Structures

## Fall 2012

## Programming Assignment 3

## Due on Wednesday, Oct 10 at 1:00 pm

Your neighbor is an agent for a group of magicians. People call her to book magicians for holidays. She would like to use her new computer to keep track of the jobs she schedules for the magicians she manages, so she hires you to write the program.

**Input**

1. The list of magician names is in text file "**`Magician.dat`**". The names are listed one per line, and have a maximum of 20 characters.

2. The list of holidays is in text file "**`Holidays.dat`**". Again, the names are in listed one per line and have a maximum of 20 characters.

3. The current schedule (retained data from the previous executions of the program) is in file "**`Schedule.dat`**". You determine the format of this file as part of your assignment.

4. The user (your neighbor the agent) inputs commands from the keyboard, in response from program prompts, as described under Command Processing below.

**Output**

1. Prompts, menus, and responses to user commands are to be written to the screen, as described in the Processing instructions below.

2. A summary of each command's transaction must be written to a text file called "**`Magic.trn`**". You may determine the format of the information in this file; it should be labeled and formatted clearly.

3. File "**`Schedule.dat`**" must be rewritten to contain the updated magician schedule information.

**Command Processing**

The program must process the commands described below. You may determine the details of the user interface; it must be relatively "friendly" and usable.

*SCHEDULE (customer) (holiday)*

Prompt the user for the name of the customer who wants to schedule a magician and for the name of the holiday to be scheduled. Check to see if there is a magician free for this holiday. You should sequence through the magicians in the order in which you read them in. If a magician is available, book the magician; then print out the name of the magician, the holiday, and the name of the customer. If a magician is not available, put the customer on a waiting list, and print out a message indicating that the customer and holiday have been put on a waiting list.

*CANCEL (customer) (holiday)*

Prompt the user for the customer name and holiday. Delete the booking of a magician for the listed holiday. Delete the reservation for that holiday. Update the schedule of the magician who was going to perform for the occasion. This may allow someone on the waiting list to be served. Sequence through the waiting list to see if someone wanted a booking on that holiday. If someone did want a magician on that holiday, schedule the booking and print a message. If this person is on the waiting list, delete the name from the waiting list.

*SIGNUP (magician)*

Prompt the user for the name of the new magician who is signing up with the agent.

*DROPOUT (magician)*

Prompt the user for the name of the magician who no longer needs the services of the agent. You must try to redistribute that magician's bookings to other magicians. If you reschedule a booking, print a message. If you can't, print out a message and add the request to the front of the waiting list.

*STATUS (magician or holiday)*

Prompt the user for the name of either a magician or holiday. Print out the appropriate schedule, appropriately formatted and labeled.

*QUIT*

Save the updated data to the files, and then terminate the program.

**Data Structures**

You need lists for storing each of the following:

1. A list of bookings for each holiday. (You may assume that there are at most ten holidays.) Each list is the schedule of one holiday. Each list element contains the name of the customer who made the booking and the name of the magician. Each list should be stored in alphabetical order, using the customer name as a key.

2. A list of bookings for each magician. (You may assume that there are at most ten magicians.) Each list is the schedule of one magician. Each list element contains the name of the customer who made the booking and the holiday. Each list should be stored in alphabetical order, using the holiday name as a key.

3. A waiting list. You need to be able to add to either end of the waiting list. You add to the front of the waiting list if you are rescheduling someone who had a booking, but lost it when a magician quit. You add to the back of the waiting list if someone requests an appointment and there are no free magicians.

**Testing**

This program is complicated. You should use both top-down and bottom-up testing. You need to execute the program more than one time. Make a hard copy of the output file ("**`Magic.trn`**") after each test run, as the program rewrites this file each time it is executed.

**Deliverables**

- Your design (object-oriented) including CRC cards for each class "**`prog3_design`**".
- Listing of the program's source code "**`prog3`**".
- A listing of any ADT(s) defined and used "**`adt3`**"
- A listing of the test driver for the ADT(s) "**`adt3_test`**"
- Copies of file "**`Magic.trn`**" from each of the final test executions.
- Implemented test plan and test drivers "**`test3`**".
- Provide all files under a compressed file (such as zip, tar, rar) where the file name is "**`prog3_LASTNAME`**" with your *last name*.
- Unless permission is obtained from the instructor, this assignment must be completed by a group of **2**.

*(This programming assignment was created by Rick Alterman.)*

# PROGRAM SCHEDULE

| *Milestone* | *Date Completed* *Planned* | *Actual* |
|---|---|---|
| Assignment received. | | |
| Requirements understood;<br>detailed specification recorded. | | |
| Top level of design complete. | | |
| All levels of top-down design complete;<br>data structures determined. | | |
| Coding complete (clean compile). | | |
| Test plan complete. | | |
| Testing complete | | |
| Program ready to turn in; all external and internal documentation complete. | | |
| Assignment turned in. | | |