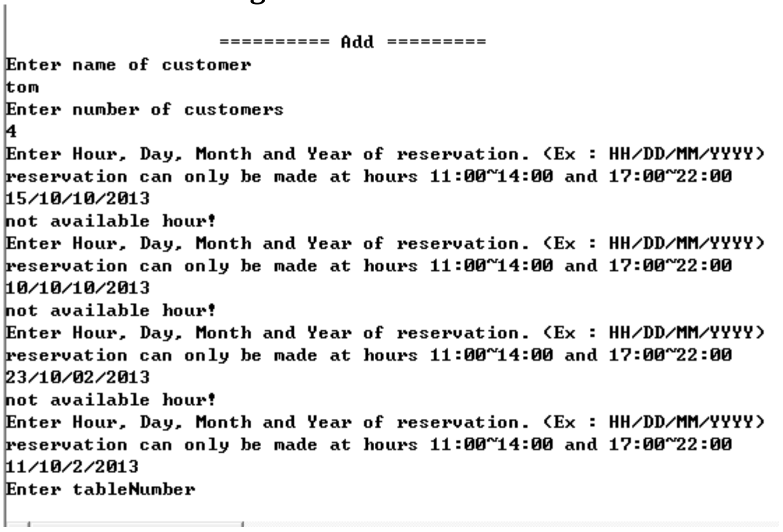
C2. Handling Errors

There are possibilities of run-time error occurring and an error can take several forms. Errors occur from calculations, input and output. The program have functions that will help the user to re-type the information again in the right manner without program crashing or having wrong information by making an error.

I will list type of errors that could be caused and how these errors were handled.

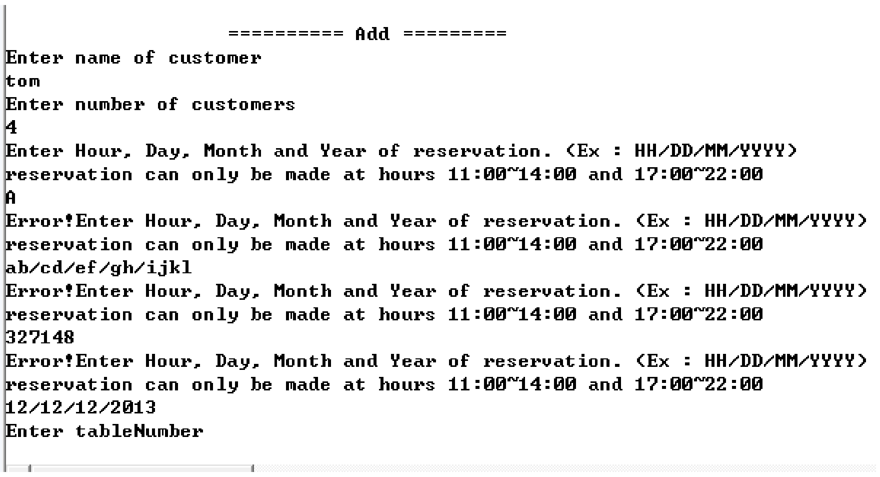
1. Out of range



```
                    ========== Add =========
Enter name of customer
tom
Enter number of customers
4
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
15/10/10/2013
not available hour!
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
10/10/10/2013
not available hour!
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
23/10/02/2013
not available hour!
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
11/10/2/2013
Enter tableNumber
```

In this case, input data was out of range. Reservation can be made only at 11:00~14:00 and 17:00~22:00. However, hour value of 15, 10 and 23 was typed by user. The program shows error message and it asks user to re-enter the data until he or she inputs data that falls in allowed range.

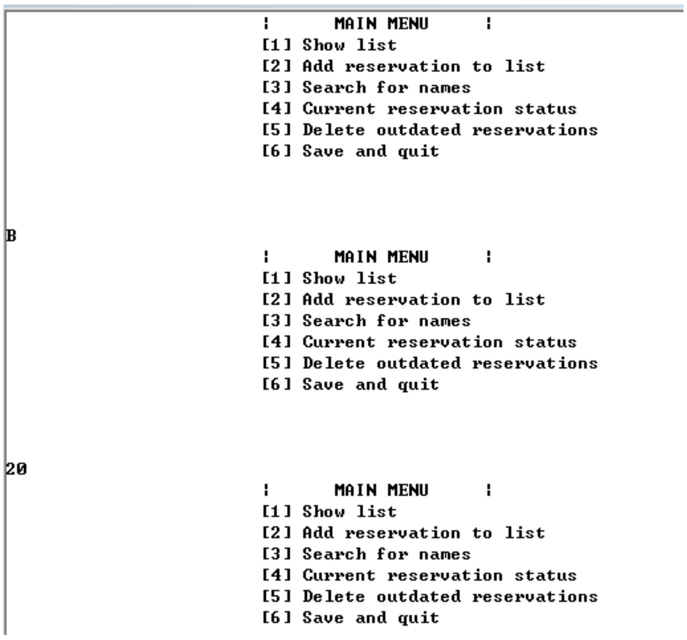
2. Data type error



```
                    ========== Add =========
Enter name of customer
tom
Enter number of customers
4
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
A
Error!Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
ab/cd/ef/gh/ijkl
Error!Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
327148
Error!Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
12/12/12/2013
Enter tableNumber
```

In this case, wrong format of data was entered by user. The program asks to enter date with format of HH/DD/MM/YYYY. However, data that doesn't follow that format was entered. After the data input, the program splits data by "/" and instantiates calendar class to convert data into milliseconds. Thus, the program will normally crash because integer array of size 4 is expected. However, it handles exception and allow user to re-enter the information.

3. Unexpected data input



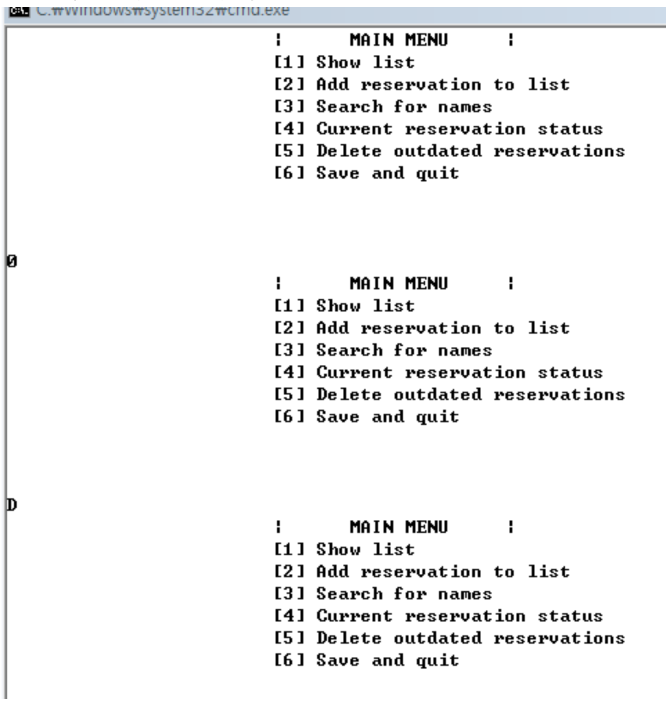
```
                          :      MAIN MENU      :
                          [1] Show list
                          [2] Add reservation to list
                          [3] Search for names
                          [4] Current reservation status
                          [5] Delete outdated reservations
                          [6] Save and quit



B
                          :      MAIN MENU      :
                          [1] Show list
                          [2] Add reservation to list
                          [3] Search for names
                          [4] Current reservation status
                          [5] Delete outdated reservations
                          [6] Save and quit



20
                          :      MAIN MENU      :
                          [1] Show list
                          [2] Add reservation to list
                          [3] Search for names
                          [4] Current reservation status
                          [5] Delete outdated reservations
                          [6] Save and quit
```

In the main menu, the program asks user to choose from six options given. It is expected that user type integer ranging from 1 to 6. However, String value and number that is not one of the options are entered by user. The program ignores the entered data and go back to main menu for user to enter acceptable data.

Stage D : Documentation

D1. Annotated hard copy of the test output

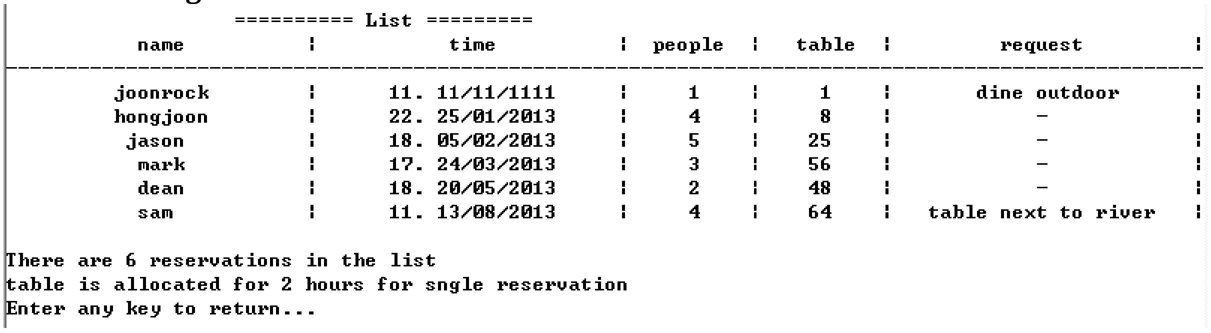
1. Main Menu



```
C:₩Windows₩system32₩cmd.exe
                          |      MAIN MENU      |
                          [1] Show list
                          [2] Add reservation to list
                          [3] Search for names
                          [4] Current reservation status
                          [5] Delete outdated reservations
                          [6] Save and quit



0
                          |      MAIN MENU      |
                          [1] Show list
                          [2] Add reservation to list
                          [3] Search for names
                          [4] Current reservation status
                          [5] Delete outdated reservations
                          [6] Save and quit



D
                          |      MAIN MENU      |
                          [1] Show list
                          [2] Add reservation to list
                          [3] Search for names
                          [4] Current reservation status
                          [5] Delete outdated reservations
                          [6] Save and quit
```

This is how main menu look like. It is the first screen that user will see and he or she can choose from listing reservations, adding reservations, editing and deleting after searching through reservations, showing reservations at particular time, deleting outdated reservations, saving and quitting the program. Program will ignore any input that isn't an option, such as 0 and "D".

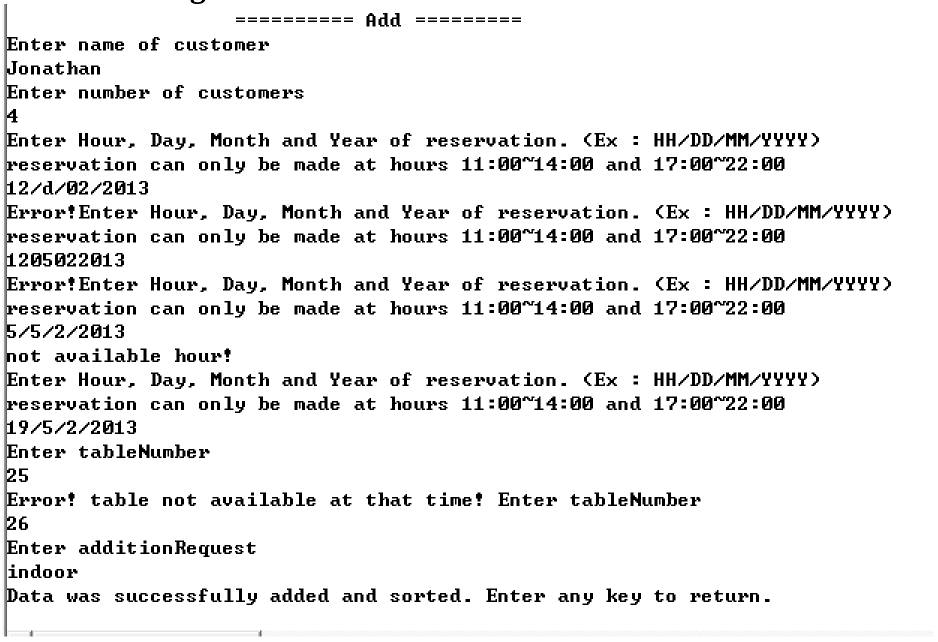
2. Listing



```
                    ========== List =========
        name          |          time           |  people  |   table   |         request          |
--------------------------------------------------------------------------------------------------
      joonrock        |      11. 11/11/1111     |    1     |     1     |       dine outdoor       |
      hongjoon        |      22. 25/01/2013     |    4     |     8     |            -             |
       jason          |      18. 05/02/2013     |    5     |    25     |            -             |
        mark          |      17. 24/03/2013     |    3     |    56     |            -             |
        dean          |      18. 20/05/2013     |    2     |    48     |            -             |
        sam           |      11. 13/08/2013     |    4     |    64     |    table next to river   |

There are 6 reservations in the list
table is allocated for 2 hours for sngle reservation
Enter any key to return...
```

This is an example of what user see when he/she enters 1 at main menu. The program will list all the reservations and it will include name, time of reservation, number of people, table number (each table in the restaurant is allocated with its unique number) and additional request. Entering any key will redirect user to main menu. It will appear as table and all information will be aligned to middle. It makes user to easily read reservation records, and it achieves one of the objectives in objective C.

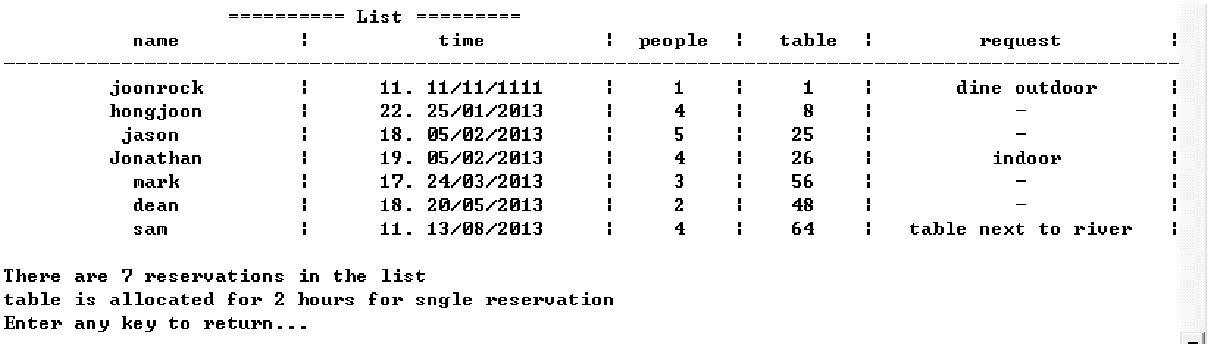
3. **Adding reservations.**



```
                  ========== Add =========
Enter name of customer
Jonathan
Enter number of customers
4
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
12/d/02/2013
Error!Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
1205022013
Error!Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
5/5/2/2013
not available hour!
Enter Hour, Day, Month and Year of reservation. (Ex : HH/DD/MM/YYYY)
reservation can only be made at hours 11:00~14:00 and 17:00~22:00
19/5/2/2013
Enter tableNumber
25
Error! table not available at that time! Enter tableNumber
26
Enter additionRequest
indoor
Data was successfully added and sorted. Enter any key to return.
```

When user input 2 in main menu, he/she will be able to add a reservation. It shows how program asks user to re-enter data when he/she enters wrong data. When user was entering table number, program displayed error message because there was reservation at table 25 at 18:00 at the same day by Jason (refer to screenshot at listing section). Because one table is allocated for two hours, two reservations couldn't be made for one table at the same time.

After adding reservation, the program automatically sorts reservation in order of time of reservation.

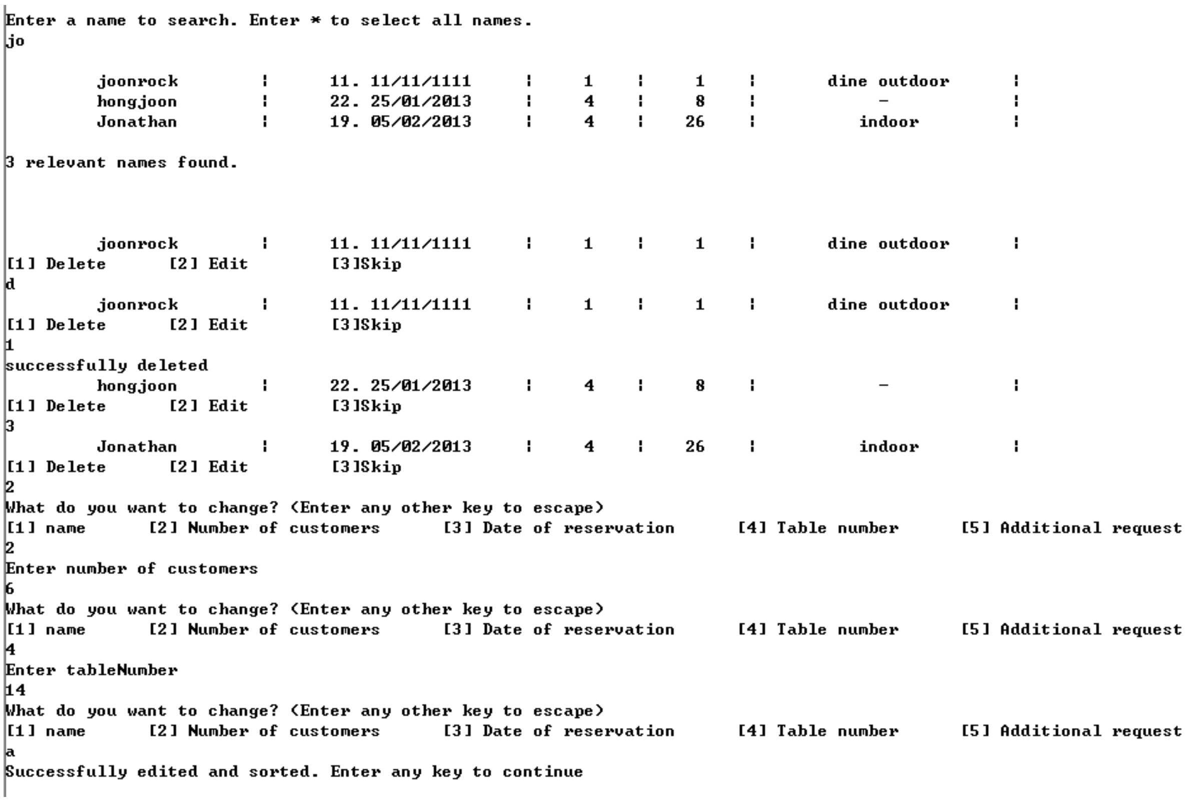


```
                  ========== List =========
         name         |          time         |  people  |  table  |         request        |
-----------------------------------------------------------------------------------------------
       joonrock       |     11. 11/11/1111    |    1     |    1    |      dine outdoor      |
       hongjoon       |     22. 25/01/2013    |    4     |    8    |           -            |
        jason         |     18. 05/02/2013    |    5     |   25    |           -            |
       Jonathan       |     19. 05/02/2013    |    4     |   26    |         indoor         |
         mark         |     17. 24/03/2013    |    3     |   56    |           -            |
         dean         |     18. 20/05/2013    |    2     |   48    |           -            |
         sam          |     11. 13/08/2013    |    4     |   64    |   table next to river  |

There are 7 reservations in the list
table is allocated for 2 hours for sngle reservation
Enter any key to return...
```

This is what appears at list menu after adding and sorting reservation. This fully achieves objective C and part of objective B

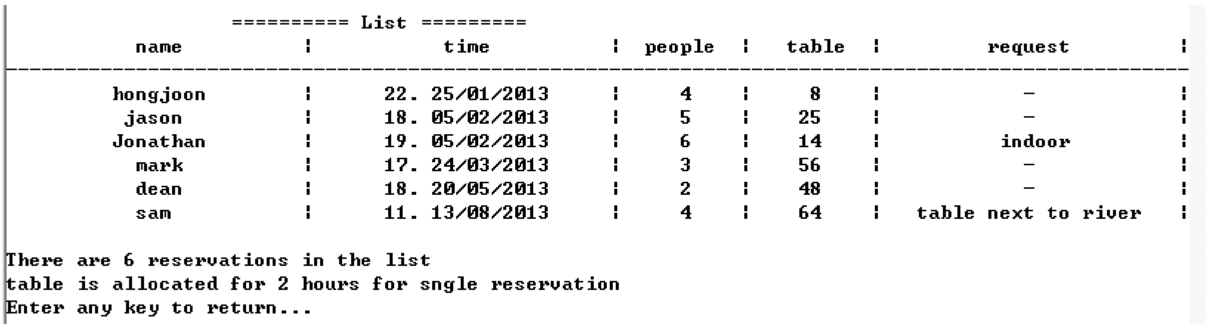
4. Searching, editing and deleting.

Clerk had to change data of reservation but she couldn't hear the customer's name clearly. So She searched with "jo" which she thought that there would be "jo" in the customer's name.



```
Enter a name to search. Enter * to select all names.
jo

        joonrock        !      11. 11/11/1111     !     1    !     1    !       dine outdoor      !
        hongjoon        !      22. 25/01/2013     !     4    !     8    !            -            !
        Jonathan        !      19. 05/02/2013     !     4    !    26    !          indoor         !

3 relevant names found.


        joonrock        !      11. 11/11/1111     !     1    !     1    !       dine outdoor      !
[1] Delete      [2] Edit        [3]Skip
d
        joonrock        !      11. 11/11/1111     !     1    !     1    !       dine outdoor      !
[1] Delete      [2] Edit        [3]Skip
1
successfully deleted
        hongjoon        !      22. 25/01/2013     !     4    !     8    !            -            !
[1] Delete      [2] Edit        [3]Skip
3
        Jonathan        !      19. 05/02/2013     !     4    !    26    !          indoor         !
[1] Delete      [2] Edit        [3]Skip
2
What do you want to change? (Enter any other key to escape)
[1] name      [2] Number of customers      [3] Date of reservation      [4] Table number      [5] Additional request
2
Enter number of customers
6
What do you want to change? (Enter any other key to escape)
[1] name      [2] Number of customers      [3] Date of reservation      [4] Table number      [5] Additional request
4
Enter tableNumber
14
What do you want to change? (Enter any other key to escape)
[1] name      [2] Number of customers      [3] Date of reservation      [4] Table number      [5] Additional request
a
Successfully edited and sorted. Enter any key to continue
```

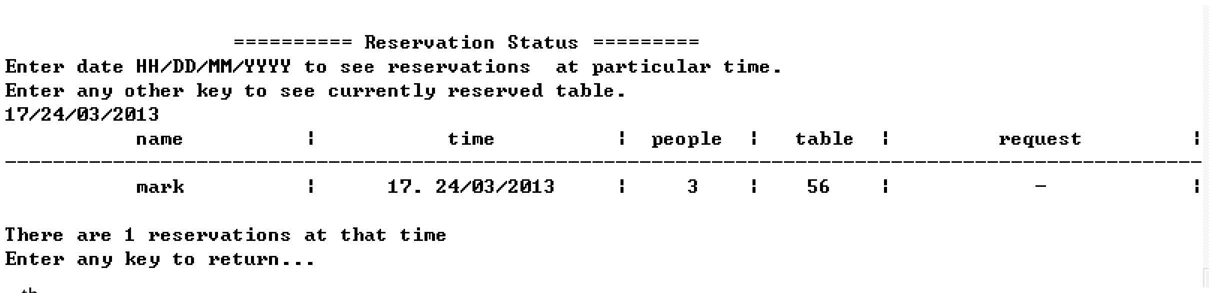
Program found three names with letters "jo" in it. Program allowed user to delete, edit or do nothing for all names that were found. User deleted joonrock's reservation because it was outdated. Program handled minor error that was made by user. He skipped hongjoon's and edited Jonathan's reservation. He changed number of customers and table. Then the user entered random key to escape from editing phase. The program sorted reservation after that.



```
                    ========== List =========
          name          !          time          !  people  !   table   !          request          !
------------------------------------------------------------------------------------------------------
        hongjoon        !      22. 25/01/2013    !     4    !     8     !             -             !
         jason          !      18. 05/02/2013    !     5    !    25     !             -             !
        Jonathan        !      19. 05/02/2013    !     6    !    14     !          indoor           !
          mark          !      17. 24/03/2013    !     3    !    56     !             -             !
          dean          !      18. 20/05/2013    !     2    !    48     !             -             !
          sam           !      11. 13/08/2013    !     4    !    64     !    table next to river    !

There are 6 reservations in the list
table is allocated for 2 hours for sngle reservation
Enter any key to return...
```

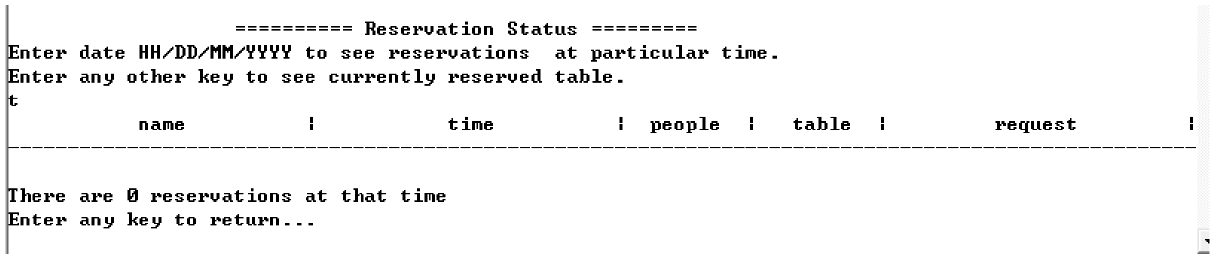
This is a listing of reservations after deleting Joonrock's and editing Jonathan's record. It achieves objective A, which is about having efficient search and also fully achieves objective B, sorting editing and deleting.

5. **Showing resrvations at particular time.**



```
                ========== Reservation Status =========
Enter date HH/DD/MM/YYYY to see reservations  at particular time.
Enter any other key to see currently reserved table.
17/24/03/2013
           name          |           time          |  people  |   table  |         request         |
-----------------------------------------------------------------------------------------------------
           mark          |      17. 24/03/2013     |     3    |    56    |            -            |

There are 1 reservations at that time
Enter any key to return...
```

4th menu is a bit about searching through reservations by dates. Date is entered and reservation at that particular time has appeared. If user enter anything else, or even if he makes an error, reservation at current time will show up.

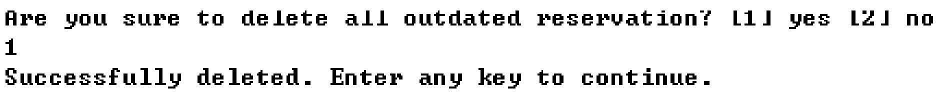


```
                ========== Reservation Status =========
Enter date HH/DD/MM/YYYY to see reservations  at particular time.
Enter any other key to see currently reserved table.
t
           name          |           time          |  people  |   table  |         request         |
-----------------------------------------------------------------------------------------------------

There are 0 reservations at that time
Enter any key to return...
```

However, there are no reservations at 23/30/01/2013

This section refers to objective A, specifically about being able to search through other detail of customer.

6. **Deleting outdated reservations.**

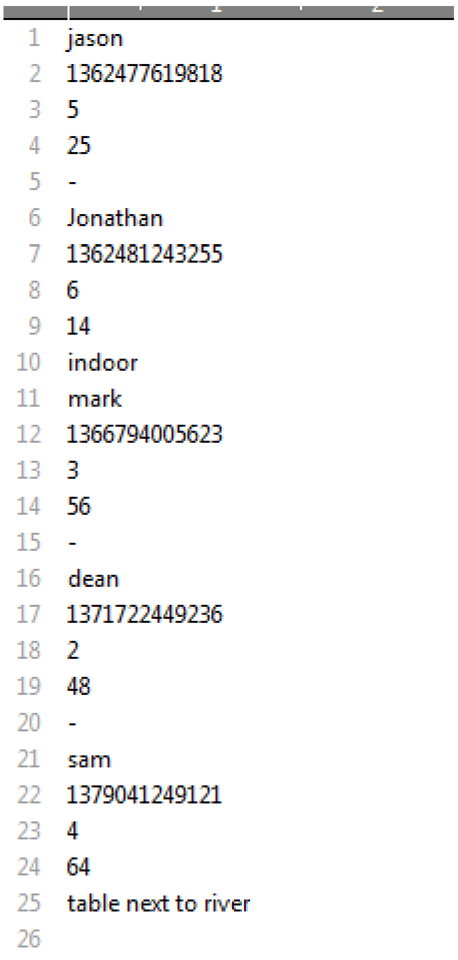


```
Are you sure to delete all outdated reservation? [1] yes [2] no
1
Successfully deleted. Enter any key to continue.
```

After processing reservation from customer, that reservation has to be deleted. However, user has to do lots of work to delete reservations one by one. Thus, 5th menu does deleting for the user.

7. Saving and quitting the program

Entering 6 in main menu will save and quit the program. The program will write all reservations into the text file “database.txt” and quit the program.



```
jason
1362477619818
5
25
-
Jonathan
1362481243255
6
14
indoor
mark
1366794005623
3
56
-
dean
1371722449236
2
48
-
sam
1379041249121
4
64
table next to river

```

This is how file looks like after saving and quitting the program. Time of reservation is saved as millisecond value.

D2. Evaluating solutions

D2.1 Meeting criterion of success at A2

Firstly, I succeeded to compile the program and made it to work. Next thing to do was to compare what the program does and what it was supposed to do, which was specified in criterion A2.

These are initial objectives of the program.

Have faster search through customer's names and their details.
Have sorting and deleting of reservation list. List should be sorted according to dates and time.
Allow user to add reservation to list, and Reservation list should be easily read.

Below describes how objectives are achieved.

1. **Objective A1**

It has fancy searching method that does not require to input exact name (letter that is included in name will find all names with it). Also, program can search reservation by time.

2. **Objective A2**

After searching data, program allows user to delete and edit reservations. Then, program automatically sorts all reservations in order of dates.

3. **Objective A3**

2nd menu of the program allows user to add reservation, and it handles input errors. Also, reservation is listed in table format and all informations are aligned to the middle. It makes user to read reservation more clearly with ease.

Therefore, I can say that the program is efficient in the manner of achieving all objectives at analysis stage. The program effectively solves the problem in JUMBO seafood restaurant in terms of objectives I stated. However, this program wasn't as efficient for it to be used in real life. I will discuss it further below.

D2.2 Limitations of the program

1. Allowance of the reservations at any moment in time

My program currently allows user to create a booking at any time of the year, including dates that have already passed. It can also add reservation for 10 years later. JUMBO seafood restaurant does not allow reservation that is more than 1 month away. So it is no point of being able to add reservation that happens in a long future.

2. Non-existance of adding list of currently occupied table

Customers don't always have to reserve restaurant to have meal there. Thus, not only reserved table will be occupied at any moment. Therefore, clerk won't be able to know which tables are empty at particular time. Also, he or she will not be able to know if restaurant is full or not.

3. Allowance of booking more than one table at the same time

My current program can only allow user to book one table only. It will basically limit allowance of number of customers, so restaurant will not be able to reserve customers in big groups.

D2.3 Improvements to the program

Program should be improved in a way that it solves limitation of the current program. Additional features might be added to the program. It includes adding feature of listing table status to see which tables are occupied at particular time. Also, the program could be improved to allow multiple tables to be reserved. To achieve this though, I would need another class and I would have to add arrays of table class other than arrays of reservation class.

Moreover, program should limit the date range of reservations. Previously, it only checked if hour of reservation is inside of data range. However, to improve the program, I should edit the program to check range of day, month and year of reservation as well.

D2.4 Evaluation of design

In my design, not all algorithms were stated in design part. As I built a program, I needed to add other algorithms that I didn't consider at design stage. Furthermore, I designed one of my algorithm and data structure for the program to be able to book more than one table at the same time, but it was not achieved. For the similar reason, my class diagram missed out lots of data fields and functions that I had in my actual program. Although I had to add lots of features in actual program than that I decided to include in design stage, there were only a few aspects which was excluded in the actual program.

To add feature of adding and viewing list of occupied table, and also to add feature of adding more than one table, the program would need another class 'Table' and Restaurant class should have two separate arrays of different user-defined data types. Also, I would have to design algorithms that connect table and reservations classes. Therefore, I would have to add additional data structures, add algorithms and change class diagrams.