Ira A. Fulton Schools of Engineering
Arizona State University

CSE 100 (C++) – Assignment #8

Maximum points: 20 pts

Topics

- 2D Array, Array of Objects (Chapter 8.10 , 8.13)
    - Object instance as an element of array.
- File I/O (Chapter 5)
    - Extract text information from a .txt file
    - ifstream object (5.12)

Your programming assignments require **individual** work and effort to be of any benefit. Every student must work independently on his or her assignments. This means that every student must ensure that neither a soft copy nor a hard copy of their work gets into the hands of another student. Sharing your assignments with others in any way is **NOT** permitted. Violations of the University Academic Integrity policy will not be ignored. The university academic integrity policy is found at https://engineering.asu.edu/integrity/

Use the following Guidelines:

- Give identifiers semantic meaning and make them easy to read (examples numStudents, gross_Pay, etc).
- Keep identifiers to a reasonably short length.
- User upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.

Important Note:

All submitted assignments must begin with the descriptive **comment block**. To avoid losing trivial points, make sure this comment header is included in every assignment you submit, and that it is updated accordingly from assignment to assignment.

```
//***********************************************************
// Name: Your name here
// Title: XXX.cpp
// Description: Short description should be typed here.
// Date: mm/dd/yyyy
//***********************************************************
```

Step 1: Dummy Functions and Display Menu (2 Pts)

This assignment has several steps. The first step is to write the sample code and submit the files to Gradescope to check whether the submission is working or not. **Don't go to the next step until you get proper points in AutoGrader**. If you cannot, meet with support members in tutoring session and/or office hours to fix the problems.

This is a project to define a class ***JetSeats*** *that modifies 2D array of Guest objects, then develop a program (Exercise1.cpp) to test the class. By reading guest's information from a file, the guests are allocated to the grid cells (seats). The test program can shift and upgrade the seats. The class has the following instance variables and functions.*

UML

```
JetSeats
- guests[6][4]  : Guest
- rows          : int
- columns       : int
+ JetSeats ()
- swapGuests(Guest&, Guest&)    : void
+ updateSeat(int, int, Guest)   : void
+ upgrade(int, char)          : bool
+ shift(char)                 : void
+ displaySeats()              : string
+ displayGuests()             : string
```

(Test1) Download and **read** **Data1.txt** **Data2.txt**, **Guest.h** and **Guest.cpp** files posted on the CANVAS page, and answer the following self-check questions about the **Guest** class (not JetSeats).

1. What are the instance variables of Guest class?
2. What do the two constructors set for the instance variables?
3. What does the getInfo() function do if the priority is equal to -1?
4. What does the getInfo() function do else case?
5. What does the equals (Guest) function do?
6. Look at the Data1.txt and Data2.txt files. Each line has the index of row, column, name of guest, and priority.

Copy the code below and make the **JetSeats.h**, **JetSeats.cpp**, and **Exercise1.cpp** files. Submit 7 files together: **JetSeats.h**, **JetSeats.cpp**, **Exercise1.cpp**, **Guest.h**, **Guest.cpp**, **Data1.txt**, and **Data2.txt**. If your score of Test1 in Gradescope is full points, then go to the next step. Otherwise, fix your problem first. If you cannot, get the help from our TA in the tutoring hours. **Don't go to the next step until you fix the first problem.**

Input and Output for Test1

```
*** Start of Program ***
[Please enter a command or ?]
?

Command Options
-----------------------------------
A: create new data
B: set the guests
C: shift the guests
D: upgrade a guest
E: display the guests (sorted)
?: display the menu
Q: quit this program

[Please enter a command or ?]
q

 ****** End of Program ******
```

Make the **JetSeats.h** below.

```
#ifndef JETSEATS_H
#define JETSEATS_H
#include <iostream>
#include "Guest.h"
using namespace std;

class JetSeats {
  private:
    Guest guests[6][4];
    int   rows;
    int   columns;
    void  swapSeats(Guest &, Guest &);
  public:
    JetSeats ( );
    void updateSeat(int, int, Guest);
    bool upgrade(int, char);
    void shift(char);
    string displayGuests();
    string displaySeats();
};
#endif
```

Make the **JetSeats.cpp** below.

```
#include "JetSeats.h"

JetSeats:: JetSeats (){ }
void JetSeats::shift(char x){}
void JetSeats::swapSeats (Guest &a, Guest &b){ }
void JetSeats::updateSeat(int r, int c, Guest g){ }
bool JetSeats::upgrade(int, char){return false;}
string JetSeats::displayGuests(){return "";}
string JetSeats::displaySeats(){return "";}
```

Make the **Exercis1.cpp** below and submit all three files to the Assignment in GradeScope

```
#include <iostream>
#include <fstream>
#include "Guest.h"
#include "JetSeats.h"
using namespace std;

void printMenu() {
  cout  <<   "\nCommand Options\n"
        <<   "-----------------------------------\n"
        <<   "A: create new data\n"
        <<   "B: set the guests\n"
        <<   "C: shift the guests\n"
        <<   "D: upgrade a guest\n"
        <<   "E: display the guests (sorted)\n"
        <<   "?: display the menu\n"
        <<   "Q: quit this program\n" << endl;
  } // end of the printMenu method

int main(){

  char command;
  string input ="";  // for file name
  ifstream fin;      // for reading file
  int num1, num2;    // for x, y inputs
  cout << "*** Start of Program ***" << endl;

  // ******************************************//
  // ** Construct JetSeats with Default Guests **
  // ******************************************//

  do
  {
     // ask a user to choose a command
      cout << "[Please enter a command or ?] " << endl;
      cin >> command;
      command = toupper(command);
      cin.ignore(20, '\n');

    switch(command){
      case 'A':
        cout <<"A:[create new data]" <<endl;
        // ********************************//
        // ****** Construct JetSeats *******
        // ********************************//
        break;
      case 'B':
        cout <<"B:[set the guests]" <<endl;
        cout << "    [Type the file name]:" << endl;
      // ********************************//
```

```
          // ** Read File and update Guests **
          // **********************************//
          break;
        case 'C':
          cout<<"C:[shift the guests] " <<endl;
          cout<<"\t[Input the direction W,A,S, or D]:"<<endl;
          break;
        case 'D':
          cout<<"D:[upgrade a guest] " <<endl;
          cout << "\t[Type Seat-ID (int and char) to upgrade]:" << endl;
          break;
        case 'E':
          cout<<"E:[display the guests (sorted)] " <<endl;
          break;
        case '?':
          printMenu();
          break;
        case 'Q':
          break;
        default:
          cout << "[Invalid input] " << endl;
          break;
      }
  } while (command != 'Q');
  cout << "\n ****** End of Program ******" << endl;
  return 0;
}
```

Step 2: Constructor and Command-A (2 Pts)

(Test2) Complete the constructor `JetSeats()` and `displaySeats()` function.

Before starting the JetSeats class, read the provided Guest.cpp, Guest.h files carefully. Without the knowledge of methods of them, it is impossible to complete this assignment.

`JetSeats()`	It instantiates a two-dimensional array `of guests`. The size is 4 columns and 6 rows, so totally 24 guests are initialized. Each `Guest` object is constructed by using the Guest's **default** constructor, `Guest()`. Two static variables, `rows` and `columns` are initialized as 6 and 4.
`string displaySeats()`	It displays the initials of Guest's name and priority at the allocated positions as well as the [seat ID]. Use the Guest's getInfo() method.

Once the two functions are implemented, go back to the main cpp file and create a default JetSeats test object just before do-loop block below. Each object is initialize as a default one.

```
// ****************************************//
// ** Construct JetSeats with Default Guests **
// ****************************************//
```

Go to the block of command-'A', and initialize the object with the user inputs. Also, add the printout code to display the history by calling `displaySeats()`.

```
        jetSeats = JetSeats();
        cout << jetSeats.displaySeats() << endl;
```

Input and Output for Test2

```
*** Start of Program ***
[Please enter a command or ?]
A
A:[create new data]
      [A]  [B]  [C]  [D]
[1]  *    *    *    *
[2]  *    *    *    *
[3]  *    *    *    *
[4]  *    *    *    *
[5]  *    *    *    *
[6]  *    *    *    *

[Please enter a command or ?]
Q

 ****** End of Program ******
```

If your scores of **Test1-2** in Gradescope are full points, then go to the next step. Otherwise, fix your problem first. If you cannot, get the help from our TA in the tutoring hours. **Don't go to the next step until you fix the first problem.**

Step 3: Read file and Update Seats (4 pts)

(Test3 & 4) Complete the `updateSeat(int col, int row, Guest g)` function.

`void updateGuest (int row, int col, Guest g)`	It reads two integers and one `Guest` object, and replaces the Guest object at [row][col] with the input Guest `g`, **only if it is equal to the default Guest** with no name and -1 priority.

Then, go to the main cpp file, and update the Command-B block to read a file name, "**Data1.txt**" and/or "**Data2.txt**" file. By accessing the data in the file, assign a Guest object at the proper position line by line. Look at the Step1 self-question 6) for the details. The code looks like below.

```
    while (!fin.eof()) {
            ...
            Guest each (...);
            seats.updateSeat(num1, num2, each);
        }
        fin.close();
        cout << seats.displaySeats() << endl;
```

The following is the result of reading the Data1.txt.

```
    [A] [B] [C] [D]
[1] M-0 G-7 *   M-5
[2] E-3 *   *   R-6
[3] *   *   *   *
[4] *   *   C-1 *
[5] *   Y-2 *   *
[6] *   *   J-4 *
```

Step 4: Shift WASD Operators (4 pts)

(Test5 & 6) Complete the `swapSeats(Guest &a, Guest &b)` function is to swap the position of guests, and the `shift(char)` function to move all guests in one direction.

`void` **`shift`** `(char command)`	It reads a direction command and shifts **all guests** to next seats in a given direction corresponding to the direction command: **up ('W'), left ('A'), down ('S') and right ('D').**
`void` **`swapGuests`** `(Guest &a, Guest &b)`	It reads two variables referring Guest and swap the guests. This method is used in the `shirt(char)` method.

Once the functions are implemented, go back to the main cpp file and update the Command-C block, which is to ask the user to type one of the direction key (W, A, S, D). Then call the shift(char direction-key) function to move all guests to the next seats.

```
    seats.shift(direction_key);
    cout << seats.displaySeats() << endl;
```

The following is a sample of `shift ('A')` to move all guest to left.

Part of Input and Output for Test 5 & 6

```
...
```

```
     [A]  [B]  [C]  [D]
[1]  M-0  G-7  *    M-5
[2]  E-3  *    *    R-6
[3]  *    *    *    *
[4]  *    *    C-1  *
[5]  *    Y-2  *    *
[6]  *    *    J-4  *

[Please enter a command or ?]
C
C:[shift the guests]
     [Input the direction W,A,S, or D]:A
     [A]  [B]  [C]  [D]
[1]  G-7  *    M-5  M-0
[2]  *    *    R-6  E-3
[3]  *         *    *
[4]  *    C-1  *    *
[5]  Y-2  *    *    *
[6]  *    J-4  *    *

…
```

If your scores of **Test#1 - #6** in Gradescope are full points, then go to the next step. Otherwise, fix your problem first. If you cannot, get the help from our TA in the tutoring hours. **Don't go to the next step until you fix the first problem.**

Step5 Upgrade the Seat (4 Pts)

Test 7, 8, 9) Complete the `upgrade(int, char)` function, and test the command-D in the main program.

`bool upgrade (int r, char c)`	It reads the seat ID (row number and column letter) and upgrades the seat to the Business Class (the first or second row seats) if there is an empty seat (occupied by default Guest). If it is upgraded successfully, it returns **true**. If all business class seats are filled, do nothing and return **false**.

Once the functions are implemented, go back to the main cpp file. Update the Command-D block to read the seat ID (integer and character), and swap the guest and one of the Empty/Default Guest in the business class (first or second row).

```
      if (seats.upgrade(num1, command)){
         cout << "*** Update was done! ***" << endl;
         cout << seats.displaySeats() << endl;
       }
       else cout << "*** Update was failed! ***" << endl;
```

Part of Input and Output for Test7

```
     [A]  [B]  [C]  [D]
```

```
[1] M-0 G-7 *   M-5
[2] E-3 *   *   R-6
[3] *   *   *   *
[4] *   *   C-1 *
[5] *   Y-2 *   *
[6] *   *   J-4 *

[Please enter a command or ?]
D
D:[upgrade a guest]
    [Type Seat-ID (int and char) to upgrade]:
4 C
*** Update was done! ***
    [A] [B] [C] [D]
[1] M-0 G-7 C-1 M-5
[2] E-3 *   *   R-6
[3] *   *   *   *
[4] *   *   *   *
[5] *   Y-2 *   *
[6] *   *   J-4 *
```

If your scores of **Test#1 – #8** in Gradescope are full points, then go to the next step. Otherwise, fix your problem first. If you cannot, get the help from our TA in the tutoring hours. **Don't go to the next step until you fix the first problem.**

Step 6: Sort the Guests by Name and display them (2 Pts)

Test 10) The last (**Hard**) step is to complete the displayGuests() function to display all guests sorted by the name of guests. Look at the example output and display the data in the same format. It is OK to use the vector<string> to store the real (not empty) guest data as a string, and add each name by comparing the alphabetic order.

`string displayGuests()`	It returns a string object, in which each line has each guest name, seat position, and priority. The list should be sorted by name (alphabetic order)

Once the functions are implemented, go back to the main cpp file. Update the Command-E block to call the functions to display the information. This command does not display the seats.

```
    cout << seats.displayGuests() << endl;
```

Part of Input and Output for Test10

```
    [A] [B] [C] [D]
```

```
[1] M-0 G-7 C-1 M-5
[2] E-3 *   *   R-6
[3] *   *   *   *
[4] *   *   *   *
[5] *   Y-2 *   *
[6] *   *   J-4 *

[Please enter a command or ?]
E
E:[display the guests (sorted)]
Chris 1C 1
Emily 2A 3
Grace 1B 7
John 6C 4
Mary 1D 5
Mike 1A 0
Roy 2D 6
Yoshi 5B 2
```

IMPORTANT

- Use only the statements that have been covered in class to date. This means you CAN use the items in Chapter 1,2, 3, 4, 5, 6, 7, 8 and 9 of textbook. **DO NOT use any other techniques not explained in class.** If in doubt, ask your TA or instructor. If you use them, only half score will be given at most. Be careful about this rule.
- If your program file does not run (compile error), then you may **lose the points for the exercise without any partial points.**

****

Submit your homework by following the instructions below:

****

- Go to the course web site (my.asu.edu), and then click on the GradeScope on CANVAS.
- Submit your all required files (Exercise1.cpp, …) together on-line at once.
- The file must compile and run as you submit it. You can confirm this by viewing your submission results in GradeScope. The **Output/Code button** at the right top corner in GradeScope helps to check if your submitted file runs correct one or not. The auto-grader will show 0 point, when your output is different from the expected one.

Important Note: You may **resubmit** as many times as you like before the deadline, but we will only mark your last submission.

NO submission history may be suspected as CHEATING!
NO LATE ASSIGNMENTS WILL BE ACCEPTED.