**University of Michigan – Dearborn**

**CIS 150 – Computer Science 1**

**FINAL PROJECT**

**Profesor Bacha**

Demetrius Johnson

[meech@umich.edu](mailto:meech@umich.edu)

April 17, 2020

**Table of Contents**

Contents

[**Component 1: Software System Design** 3](#_Toc37993871)

[**Component 2: Software Documentation – see** FINAL PROJECT CIS-150 WINTER 2020 - MEECH.cpp” & other .CPP files attached to submission 4](#_Toc37993872)

[**SOURCECODE: MAIN();** 4](#_Toc37993873)

[**SOURCECODE: CLASS();** 4](#_Toc37993874)

[**SOURCECODE: HEADERFILE FOR CLASS.H** 4](#_Toc37993875)

[**Component 3: Output Snippets** 4](#_Toc37993876)

[**Component 4: EXTRA CREDIT – COMPLETED – WILL USE VECTORS AND CLASSES/OBJECTS INSTEAD OF ARRAYS TO ACHIEVE THIS** 16](#_Toc37993877)

# **Component 1: Software System Design –Primary components**

# **Component 2: Software Documentation – see** FINAL PROJECT CIS-150 WINTER 2020 - MEECH.cpp” & other .CPP files attached to submission

# **SOURCECODE: MAIN();**

// FINAL PROJECT CIS-150 WINTER 2020 - MEECH.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

//\*\*\*\*\*\*\*\*\*\*\*\*EXTRA CREDIT PORTION COMPLETED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HAVE BUILT MY PROGRAM AROUND ANY NUMBER OF ROWS AND LETTERS IN ROW AND EVEN MIXTURES\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*YOU CAN USE ANY INPUT FILE OF THE SAME FORMAT. JUST END FILE WITHOUT A NEWLINE, AND ENSURE YOU HAVE EACH ROW START WITH ONLY 1 NUMBER AND END WITH A LETTER, THEN A NEWLINE (EXCPET FOR LAST ROW).

//ROWS DO NOT HAVE TO HAVE SAME NUMBER OF SEATS/LETTERS.

// Purpose: Airline Seat Reservation/Cancellation Program -- Airliner of any size seating (EXTRA CREDIT)

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

#include "RowLetters.h"

#include <iostream>

#include <fstream>

#include <vector>

#include <iomanip>

#include <string>

#include <sstream>

using namespace std;

//-------------------------------FUNCTION DECLARATIONS----------------------------------------//

void readAircraftSeatingInfo(ifstream& ifs, vector<RowLetters>& seatingVector);

void writeAircraftSeatingInfo(ofstream& ofs, vector<RowLetters>& seatingVector);

void X\_ammendAircraftSeatingInfo(string userEntry, vector<RowLetters>& seatingVector);

void Char\_ammendAircraftSeatingInfo(string userEntry, vector<RowLetters>& seatingVector);

void displayMenu();

void mainMenu();

void displaySeatingChart(vector<RowLetters>& seatingVector);

void displayTableHeader();

//------------------------------------FUNCTION DELCARATIONS------------------------------------//

int main()

{

int userChoice, stringPosition = 0, errorChecker = 0;

string userSeatEntry;

string airCraftSeatingInfoFile = "chartIn.txt";

vector <RowLetters> seatingVector;

ifstream ifs;

ofstream ofs;

char test;

//------INTIAL FILE OPEN/CLOSE------//

ifs.open(airCraftSeatingInfoFile); //open seating info data file necessarry to run the program & ensure file is found/opens

if (!ifs.is\_open()) {

cout << "There was a problem accessing the input data file to run this program.\n";

cout << "Please ensure the required data file is in the correct file directory.\n";

cout << "Program will exit. File not found or corrupt. Thank you.\n";

return 0;

}

readAircraftSeatingInfo(ifs, seatingVector); //read RowLetters info into vector

ifs.close(); //close file after all the data has been successfully read into the RowLetterss vector

//------INTIAL FILE OPEN/CLOSE------//

mainMenu(); //display menu & process input

cin >> userChoice; //initial userChoice entry

do {

switch (userChoice) {

case 1: //display seat chart\*\*\*COMPLETED

cout << "\n\nYou have selected menu option 1>> \n\n";

system("pause");

cout << endl;

displaySeatingChart(seatingVector); //USE FUNCTION TO DISPLAY OBJECTS CURRENTLY STORED IN VECTOR WHICH WILL REFLECT INPUT FILE SEATING DATA

displayMenu();

break;

case 2: //reserve seat

cout << "\n\nYou have selected menu option 2>> \n\n";

system("pause");

cout << "Please enter a seat reservation (i.e. 8C) or enter only a letter to return to the menu: ";

cin >> userSeatEntry;

for (stringPosition = 0; stringPosition < userSeatEntry.size(); stringPosition++) { //capitalize the letter in userSeatEntry

if (isalpha(userSeatEntry.at(stringPosition))) {

userSeatEntry.at(stringPosition) = toupper(userSeatEntry.at(stringPosition));

}

}

for (stringPosition = 0; stringPosition < userSeatEntry.size(); stringPosition++) { //error correction in case of invalid entry

if (isdigit(userSeatEntry.at(stringPosition))) {

errorChecker++;

}

}

if (errorChecker == userSeatEntry.size()) {

cout << "\n\*\*\*Invalid Entry. Returning to Menu.\*\*\*\n";

errorChecker = 0; //reset error checker

system("pause");

displayMenu();

break;

}

else{errorChecker = 0;}

if (!isdigit(userSeatEntry.at(0))) { //return to menu if only a character is provided

cout << "\n\nYou have selected to return to the menu >>\n\n";

system("pause");

displayMenu();

break;

}

cout << endl;

cout << "\*\*\*\*\*>>>seat " << userSeatEntry << " selected. ";

system("pause");

ofs.open(airCraftSeatingInfoFile); //open seating info data file necessarry to run the program & ensure file is found/opens

if (!ofs.is\_open()) {

cout << "There was a problem accessing the output data file to run this program.\n";

cout << "Please ensure the required data file is in the correct file directory.\n";

cout << "Program will exit. File not found or corrupt. Thank you.\n";

return 0;

}

X\_ammendAircraftSeatingInfo(userSeatEntry, seatingVector);

writeAircraftSeatingInfo(ofs, seatingVector);

ofs.close(); //close file after successfully updating seating chart with user reservation information

displayMenu();

break;

case 3: //cancel reservation

cout << "\n\nYou have selected menu option 3>> \n\n";

system("pause");

cout << "Please enter your currently reserved seat for CANCELLATION (i.e. 8C) or enter only a letter to return to the menu: ";

cin >> userSeatEntry;

for (stringPosition = 0; stringPosition < userSeatEntry.size(); stringPosition++) { //capitalize the letter in userSeatEntry

if (isalpha(userSeatEntry.at(stringPosition))) {

userSeatEntry.at(stringPosition) = toupper(userSeatEntry.at(stringPosition));

}

}

for (stringPosition = 0; stringPosition < userSeatEntry.size(); stringPosition++) { //error correction in case of invalid entry

if (isdigit(userSeatEntry.at(stringPosition))) {

errorChecker++;

}

}

if (errorChecker == userSeatEntry.size()) {

cout << "\n\*\*\*Invalid Entry. Returning to Menu.\*\*\*\n";

errorChecker = 0; //reset error checker

system("pause");

displayMenu();

break;

}

else { errorChecker = 0; }

if (!isdigit(userSeatEntry.at(0))) { //return to menu if only a character is provided

cout << "\n\nYou have selected to return to the menu >>\n\n";

system("pause");

displayMenu();

break;

}

cout << endl;

cout << "\*\*\*\*\*>>>seat " << userSeatEntry << " selected for CANCELLATION. ";

system("pause");

ofs.open(airCraftSeatingInfoFile); //open seating info data file necessarry to run the program & ensure file is found/opens

if (!ofs.is\_open()) { //open file for amending

cout << "There was a problem accessing the output data file to run this program.\n";

cout << "Please ensure the required data file is in the correct file directory.\n";

cout << "Program will exit. File not found or corrupt. Thank you.\n";

return 0;

}

Char\_ammendAircraftSeatingInfo(userSeatEntry, seatingVector);

writeAircraftSeatingInfo(ofs, seatingVector);

ofs.close(); //close file after successfully updating seating chart with user reservation information

displayMenu();

break;

default:

cout << "\n\nYou have selected to exit the program. Thank you for seating with us(:\n\n";

system("pause");

return 0;

}

cin >> userChoice;

} while (userChoice == 1 || userChoice == 2 || userChoice == 3);

cout << "\n\nYou have selected to exit the program. Thank you for seating with us(:\n\n";

system("pause");

return 0;

}

//------------------------------------FUNCTION DEFINITIONS BELOW THIS LINE------------------------------------//

// Purpose: READ FILE INFORMATION INTO VECTOR

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void readAircraftSeatingInfo(ifstream& ifs, vector<RowLetters>& seatingVector) {

RowLetters currentRow;

string dataLineFromInputFile;

string rowLetters;

stringstream ss; //temporary storage variables for ss parsing to store data in tempRowLetters object

int rowNum; //^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

int fileCheckDataLineCounter = 0;

while (!ifs.eof())

{

getline(ifs, dataLineFromInputFile, '\n'); //read other inputs on a given line using getline

ss << dataLineFromInputFile; //output dataline to string stream

ss >> rowNum; //input from string stream into variables

getline(ss, rowLetters);

ss.clear(); //clear ss buffer for usage in the next iteration of the loop

dataLineFromInputFile.clear(); //clear out string

//use object.method() for passing variables into the object through the method------------

currentRow.setRowNum(rowNum);

currentRow.setLettersInRow(rowLetters);

//------------------------------------------------------------------------------------------

seatingVector.push\_back(currentRow); //add the object to the vector

fileCheckDataLineCounter = seatingVector.size(); //set dataline counter to RowLetters vector size to ensure 50 lines of data is read

}

cout << "\nFileChecker::number of datalines read from program files is: " << fileCheckDataLineCounter << endl;

}

// Purpose: WRITE TO AND AMMEND AIRLINER SEAT INFORMATION FILE

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void writeAircraftSeatingInfo(ofstream& ofs, vector<RowLetters>& seatingVector){

int elementPosition;

for (elementPosition = 0; elementPosition < (seatingVector.size() - 1); elementPosition++) { //do vector size - 1 to take last desired output out of the for loop

//

//

ofs << right << setw(5) << seatingVector.at(elementPosition).getRowNum() << setw(3) //

<< seatingVector.at(elementPosition).getLettersInRow() << endl; //

//

} ///////

ofs << seatingVector.at(elementPosition).getRowNum() //

<< seatingVector.at(elementPosition).getLettersInRow(); //got rid of final endl from from loop so that output file does not have an unecsessary whitespace character

}

// Purpose: AMMEND VECTOR STORING INPUT AND OUTPUT FILE DATA WITH AN X FOR SEAT RESERVATIONS

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void X\_ammendAircraftSeatingInfo(string userEntry, vector<RowLetters>& seatingVector) {

string userNumber;

char userChar;

int userNumberInt, noNumberMatch = 0, noLetterMatch;

int stringPosition;

stringstream ss;

for (stringPosition = 0; stringPosition < userEntry.size(); stringPosition++) { //put character from userEntry into user Char and capitalize the letter

if (isalpha(userEntry.at(stringPosition))) {

userEntry.at(stringPosition) = toupper(userEntry.at(stringPosition));

userChar = userEntry.at(stringPosition);

break;

}

}

for (stringPosition = 0; stringPosition < userEntry.size(); stringPosition++) { //put all digits from userEntry into userNumber string

if (isdigit(userEntry.at(stringPosition))) {

userNumber.push\_back(userEntry.at(stringPosition));

}

}

ss << userNumber;

ss >> userNumberInt; //convert usernumber string variable into an integer variable using stringstream

ss.clear();

userNumber.clear();

for (int elementPosition = 0; elementPosition < seatingVector.size(); elementPosition++) {

if (userNumberInt == seatingVector.at(elementPosition).getRowNum()) {

noLetterMatch = 0;

//The below for statement reaches 1)into seating vector 2) into an object 3) into a string private member of that object 4) into the string position of that object's private data member

for (int objectStringPosition = 0; objectStringPosition < seatingVector.at(elementPosition).getLettersInRow().size(); objectStringPosition++) {

if (userChar == seatingVector.at(elementPosition).getLettersInRow().at(objectStringPosition)) {

seatingVector.at(elementPosition).exchangeLetterInRow(objectStringPosition, "X"); //replace character in string from object inside of vector

cout << "\nYou have successfully reserved seat " << userEntry << endl << endl;

system("pause");

return;

}

else { noLetterMatch++; } //if noMatches variable increments each time during for-loop iteration, it means no matches found

if(noLetterMatch == seatingVector.at(elementPosition).getLettersInRow().size()){

cout << "\n\nNo seat available (seat is reserved), or invalid seat number.\nPlease check seating chart for available seats and try again.\n\n";

system("pause");

return;

}

}

}

else { noNumberMatch++; }

if(noNumberMatch == seatingVector.size()){

cout << "\n\nNo seat available (seat is reserved), or invalid seat number.\nPlease check seating chart for available seats and try again.\n\n";

system("pause");

return;

}

}

}

// Purpose: AMMEND VECTOR STORING INPUT AND OUTPUT FILE DATA WITH THE USER INPUT CHARACTER FOR SEAT CANCELLATIONS

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void Char\_ammendAircraftSeatingInfo(string userEntry, vector<RowLetters>& seatingVector) {

string userNumber, userCharString;

char userChar;

int userNumberInt, noNumberMatch = 0, noLetterMatch;

int stringPosition;

stringstream ss;

for (stringPosition = 0; stringPosition < userEntry.size(); stringPosition++) { //put character from userEntry into user Char and capitalize the letter

if (isalpha(userEntry.at(stringPosition))) {

userEntry.at(stringPosition) = toupper(userEntry.at(stringPosition));

userChar = userEntry.at(stringPosition);

userCharString.push\_back(userChar); //use a string to contain the userChar for using in the string.find() function which needs string parameter

break;

}

}

for (stringPosition = 0; stringPosition < userEntry.size(); stringPosition++) { //put all digits from userEntry into userNumber string

if (isdigit(userEntry.at(stringPosition))) {

userNumber.push\_back(userEntry.at(stringPosition));

}

}

ss << userNumber;

ss >> userNumberInt; //convert usernumber string variable into an integer variable using stringstream

ss.clear();

userNumber.clear();

for (int elementPosition = 0; elementPosition < seatingVector.size(); elementPosition++) {

if (userNumberInt == seatingVector.at(elementPosition).getRowNum()) {

noLetterMatch = 0;

//The below for statement reaches 1)into seating vector 2) into an object 3) into a string private member of that object 4) into the string position of that object's private data member

for (int objectStringPosition = 0; objectStringPosition < seatingVector.at(elementPosition).getLettersInRow().size(); objectStringPosition++) {

if ('X' == seatingVector.at(elementPosition).getLettersInRow().at(objectStringPosition)) {

seatingVector.at(elementPosition).exchangeLetterInRow(objectStringPosition, userCharString); //replace character in string from object inside of vector

cout << "\nYou have successfully CANCELLED your seat reservattion at seat >> " << userEntry << endl << endl;

system("pause");

return;

}

else { noLetterMatch++; } //if noMatches variable increments each time during for-loop iteration, it means no matches found as it equals counter variable size

if (noLetterMatch == seatingVector.at(elementPosition).getLettersInRow().size()) {

cout << "\n\nInvalid seat number.\nPlease check seating chart and try again.\n\n";

system("pause");

return;

}

}

}

else { noNumberMatch++; }

if (noNumberMatch == seatingVector.size()) {

cout << "\n\nInvalid seat number.\nPlease check seating chart and try again.\n\n";

system("pause");

return;

}

}

}

//------DISPLAY MENUS--------------//

// Purpose: DISPLAY REPEATED MENU WITH USER OPTIONS

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void displayMenu()

{

cout << "------------------------Menu--------------------------\n\n";

cout << "1. Display Seat Chart\n";

cout << "2. Reserve Seat //seat notation is NumberLetter (i.e. 4C) \n";

cout << "3. Cancel Reservation\n";

cout << "4. Quit\n\n";

cout << "--------------------------------------------------\n\n";

cout << "Please Enter Your Choice(1 - 4) : ";

}

// Purpose: DISPLAY INTIAL MENU WITH PURPOSE AND USER OPTIONS

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void mainMenu()

{

cout << "---------------------------------WELCOME TO THE AIRLINE SEAT RESERVATION PROGRAM-------------------------\n"

<< "This program allows the user to view, reserve, or cancel seat reservations of the current flight.\n\n";

cout << "------------------------Menu--------------------------\n";

cout << "Seat selection notation is a NUMBER followed by a LETTER. For example: 7B, or 1A.\n\n";

cout << "1. Display Seat Chart\n";

cout << "2. Reserve Seat\n";

cout << "3. Cancel Reservation\n";

cout << "4. Quit\n\n";

cout << "--------------------------------------------------\n\n";

cout << "Please Enter Your Choice(1 - 4) : ";

}

// Purpose: DISPLAY SEATING CHART OF AIRCRAFT OF CURRENTLY UPDATED SEATING CHART VECTOR THAT CONTAINS THE INFORMATION

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void displaySeatingChart(vector<RowLetters>& seatingVector) {

int spaceEveryNineRows = 0;

//Display seating chart:

for (int elementPosition = 0; elementPosition < seatingVector.size(); elementPosition++) {

cout << right << setw(5) << seatingVector.at(elementPosition).getRowNum() << setw(3)

<< seatingVector.at(elementPosition).getLettersInRow() << endl;

if (elementPosition == spaceEveryNineRows + 8) {

cout << endl;

spaceEveryNineRows += 10;

}

}

}

// Purpose: USED FROM LAB 11 AS A REFERENCE FOR USING IOMANIP

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void displayTableHeader()

{

//Display table header:

cout << endl;

cout << right << setw(20) << "RowLetters Abr." << setw(20) << "Name" << setw(20) << "Capital"

<< setw(20) << "Population" << setw(11) << "Year" << setw(15) << "#Congress" << endl;

cout << endl;

}

# **SOURCECODE: CLASS();**

// Purpose: Airline Seat Reservation/Cancellation Program -- Airliner of any size seating (EXTRA CREDIT)

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

//CLASS USED TO STORE FILE DATA

#include "RowLetters.h"

#include <iostream>

#include <vector>

#include <iomanip>

#include <string>

using namespace std;

RowLetters::RowLetters() {

rowNumber = 0;

lettersInRow = "Row Number and Letters not available";

}

void RowLetters::setRowNum(int inputNumber) {

rowNumber = inputNumber;

}

int RowLetters::getRowNum() {

return rowNumber;

}

//THE REST OF FUNCTIONS ARE SIMPLE/HEADER NOT NECESSARRY

void RowLetters::setLettersInRow(string letters) {

lettersInRow = letters;

}

string RowLetters::getLettersInRow() {

return lettersInRow;

}

// Purpose: USED TO REPLACE A CHARACTER IN THE STRING OF THE OBJECT STORED IN THE VECTOR USED IN MAIN

// Author: Demetrius E. Johnson

// Creation Date: 4-15-20

// Last Modification Date: 4-17-20

void RowLetters::exchangeLetterInRow(int stringPosition, string exchangeChar) {

lettersInRow.replace(stringPosition, 1, exchangeChar);

}

//TEST FUNCTIONS --NOT NEEDED--ONLY CONSIDERED FOR USE DURING BUILD OF MY FORMAT --KEEPING FOR FUTURE REFERENCE/SELF STUDY

string& RowLetters::getLettersInRowReference() {

return lettersInRow;

}

int& RowLetters::getRowNumReference() {

return rowNumber;

}

# **SOURCECODE: HEADERFILE FOR CLASS.H**

#include <iostream>

#include <vector>

#include <iomanip>

#include <string>

using namespace std;

#ifndef ROWLETTERS\_H

#define ROWLETTERS\_H

class RowLetters {

private:

int rowNumber;

string lettersInRow;

public:

RowLetters();

void setRowNum(int);

int getRowNum();

void setLettersInRow(string);

string getLettersInRow();

void exchangeLetterInRow(int, string);

//TEST FUNCTIONS ---NOT USED--

string& getLettersInRowReference();

int& getRowNumReference();

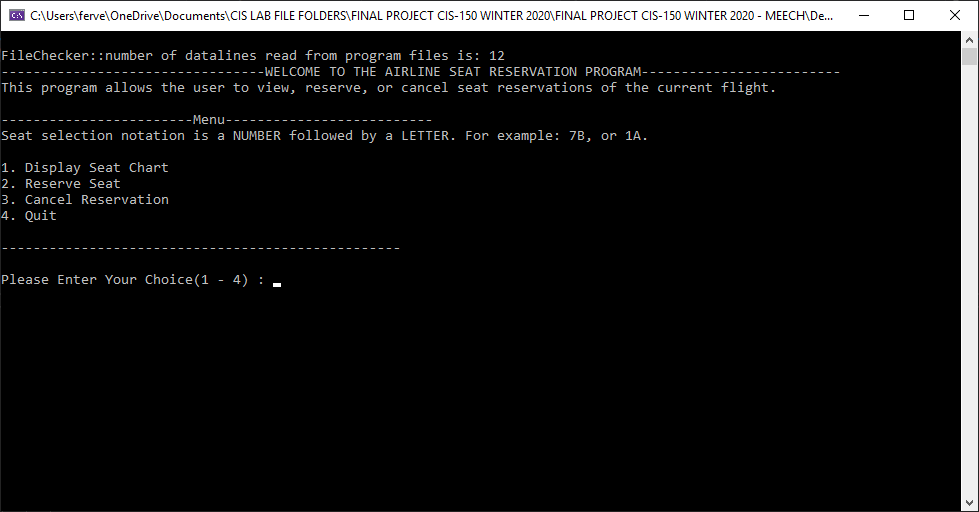
};

#endif

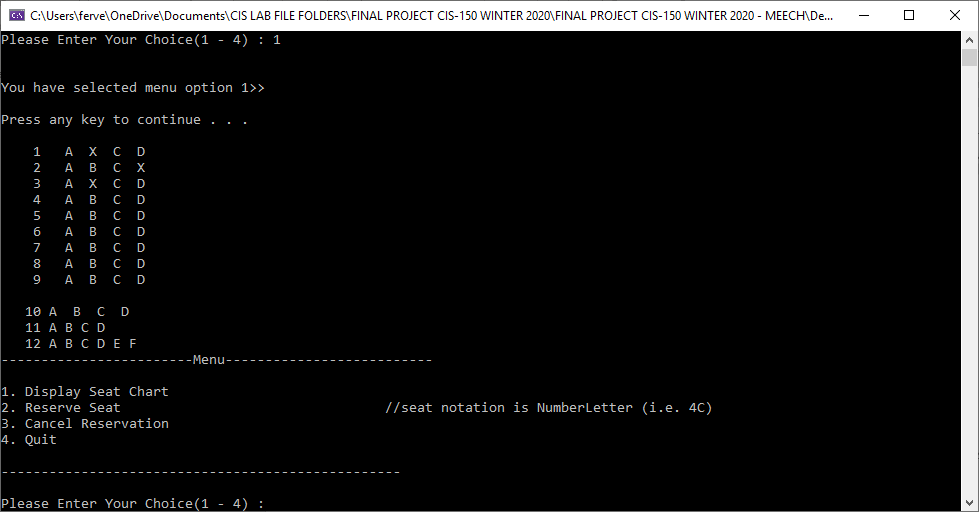
# **Component 3: Output Snippets**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Actual Output | Test Pass / Fail |
| 1 | valid | Display start of program | See screenshot | See screenshot | pass |
| 2 | valid | Display seating availability | Option 1 selected | See screenshot | pass |
| 3 | valid | Display output of selecting option 2 | Option 2 selected; reserve seat 8A | See screenshot | pass |
| 4 | valid | Display option 3 | Option 3 selected; cancel seat 1B | See screenshot | pass |
| 5 | valid | Display updated seat table showing results of tests 3 and 4 | Option 1 selected;  See screenshot | See screenshot | pass |
| 6 | valid | Reserve a seat in a double-digit row | Reserved seat 12F; see screenshot | See screenshot | pass |
| 7 | valid | Cancel a seat in a double-digit row | Cancelled seat 12F; see screenshot | See screenshot | pass |
| 8 | valid | Exit the program from first userChoice | Option 4 selected | See screenshot | pass |
| 9 | valid | Exit the program after userChoice has been manipulated more than once | Option 4 selected | See screenshot | pass |
| 10 | valid | EXTRA CREDIT  Changed file size for numrows and numseats in row | Display menu; option 1 selected | See screenshot | pass |
| 11 | valid | Reserve seat 6C in new aircraft | Option 2 selected | See screenshot | pass |
| 12 | valid | Cancel seat 6C in new aircraft | Option 3 selected | See screenshot | pass |
| 13 | valid | EXTRA CREDIT  Changed file size for numrows and numseats in row to DOUBLE DIGIT Airliner | Option 1 selected; see screenshot | See screenshot | pass |
| 14 | valid | Reserve seat 23A in new aircraft | Option 2 selected | See screenshot | pass |
| 15 | valid | Cancel seat 23A in new aircraft | Option 3 selected | See screenshot | pass |
| 16 | valid | Show a seat is already reserved | Option 2 | See screenshot | pass |
| 17 | valid | Show a user is trying to cancel an invalid seat number | Option 3 | See screenshot | pass |
|  |  | MAIN(); |  |  |  |

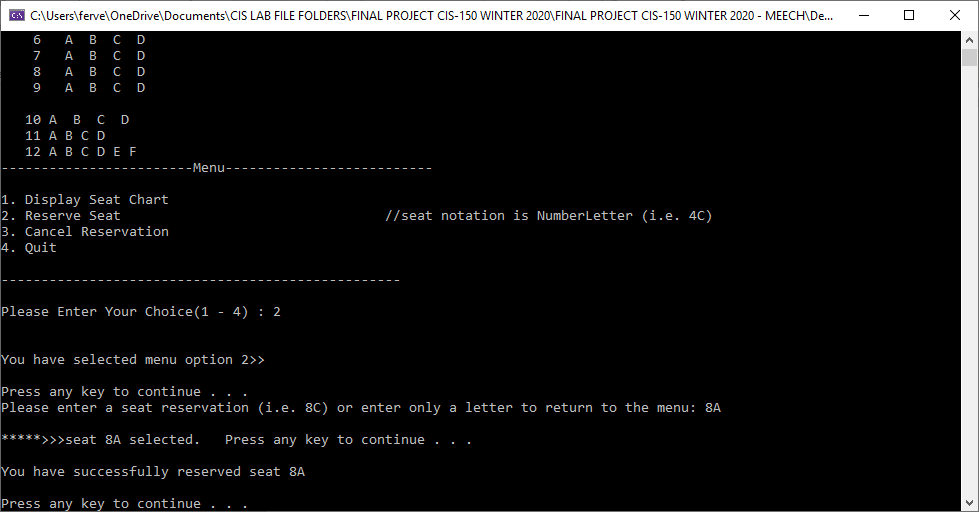
TEST 1



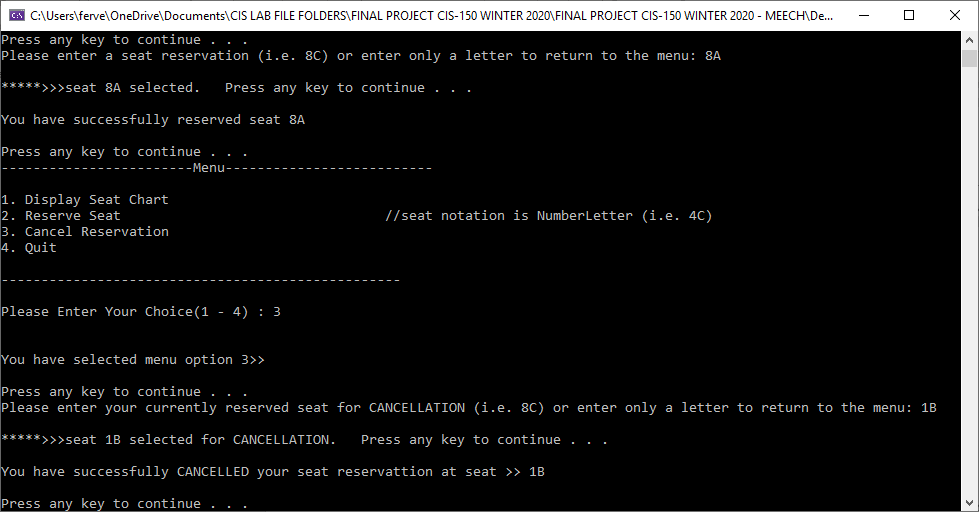
TEST 2



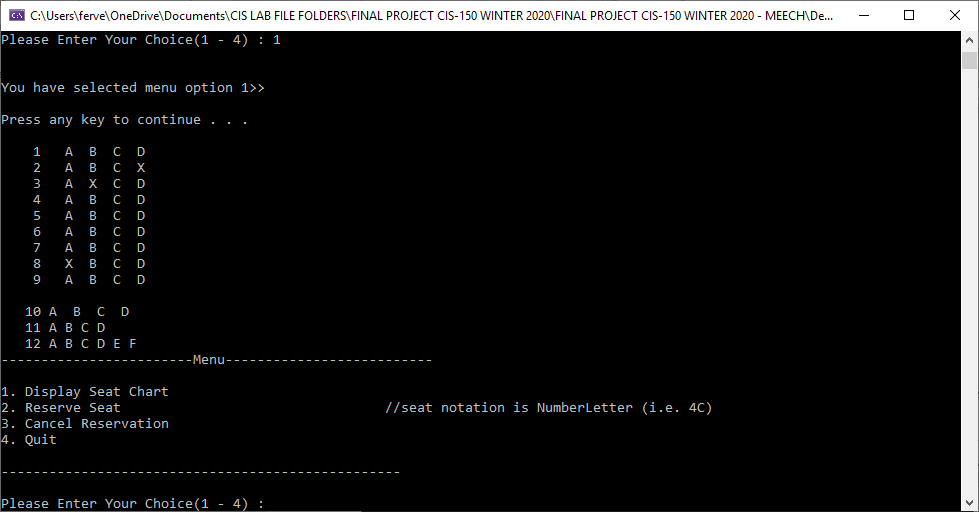
TEST 3



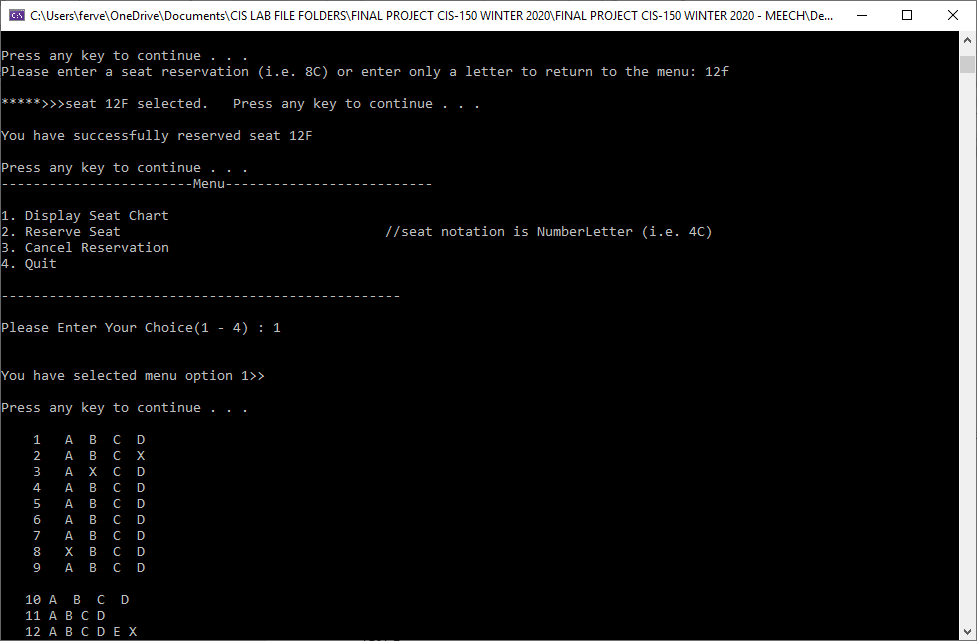
TEST 4



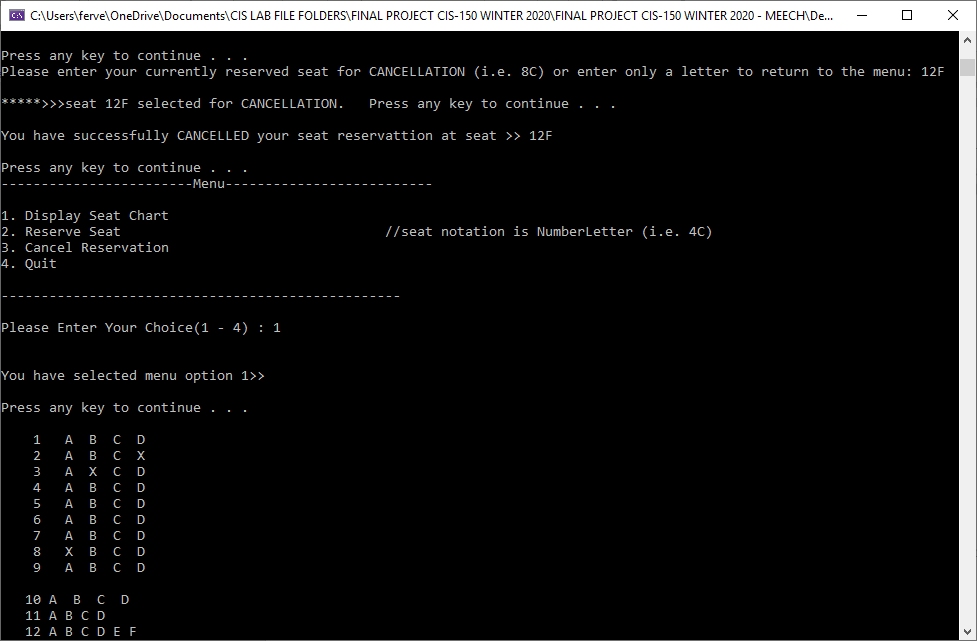
TEST 5



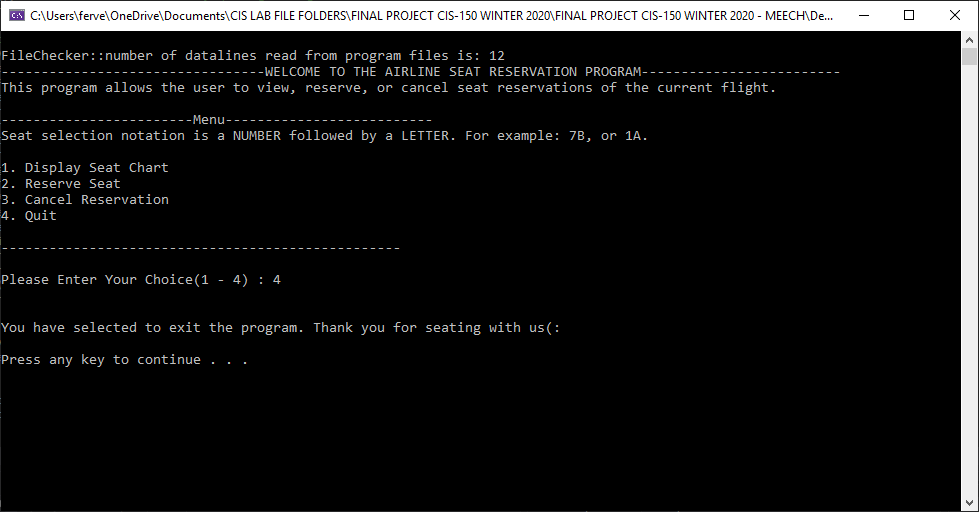
TEST 6



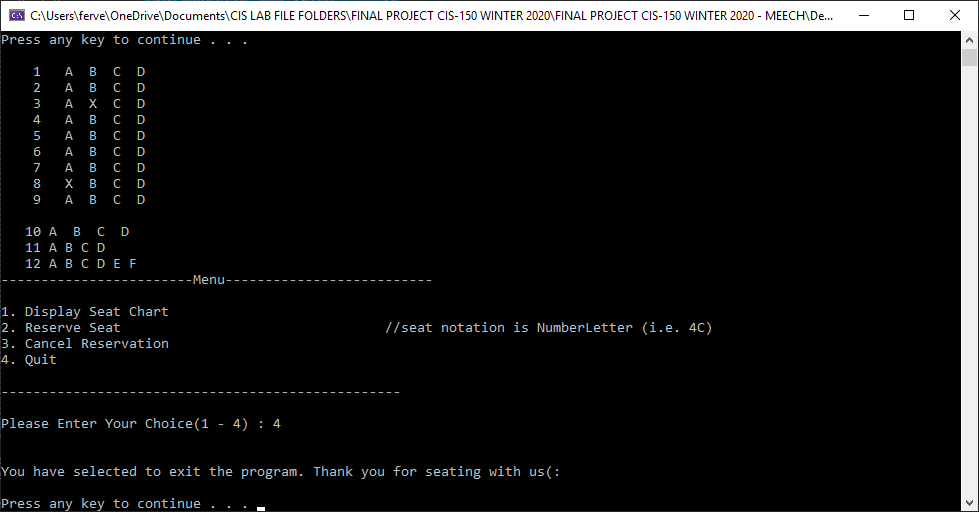
TEST 7



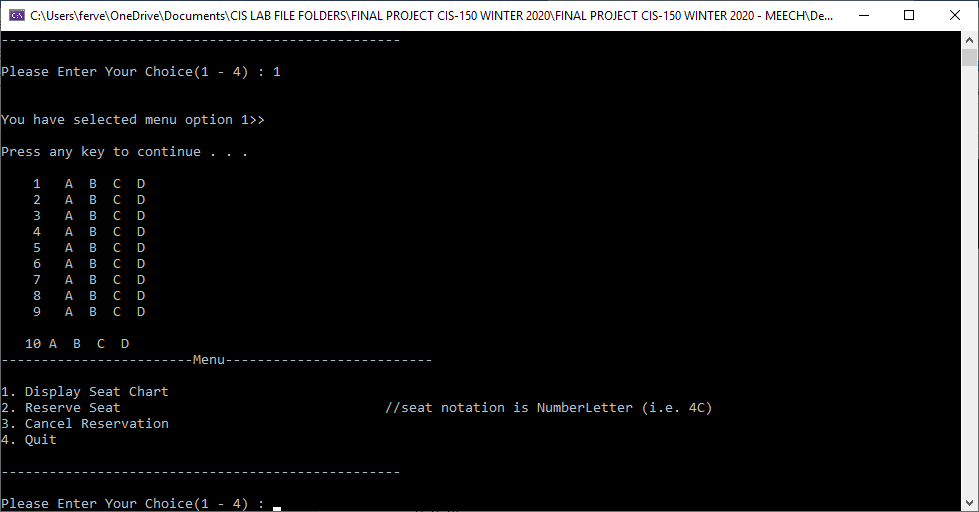
TEST 8



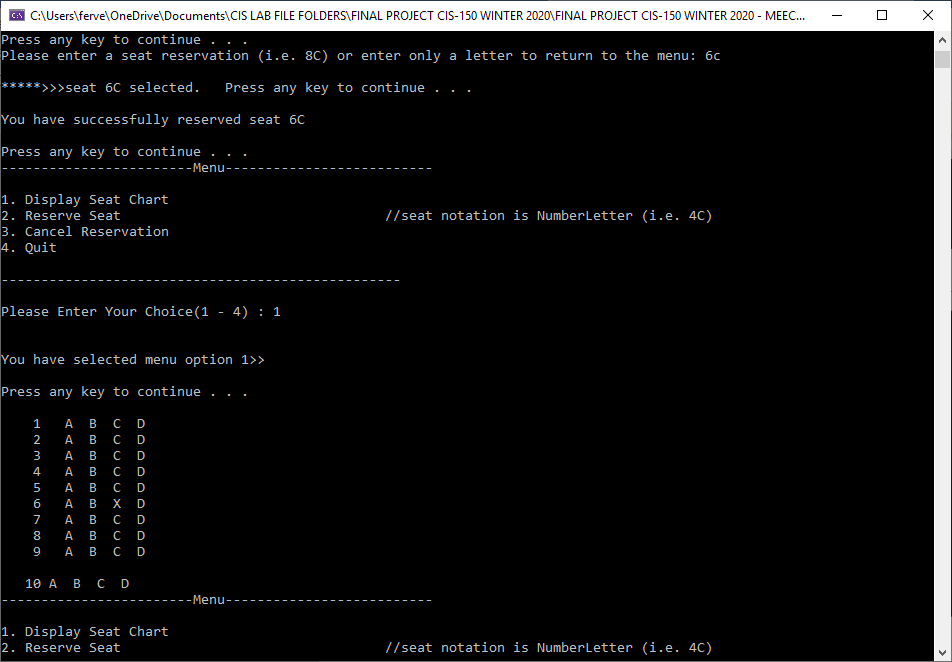
TEST 9



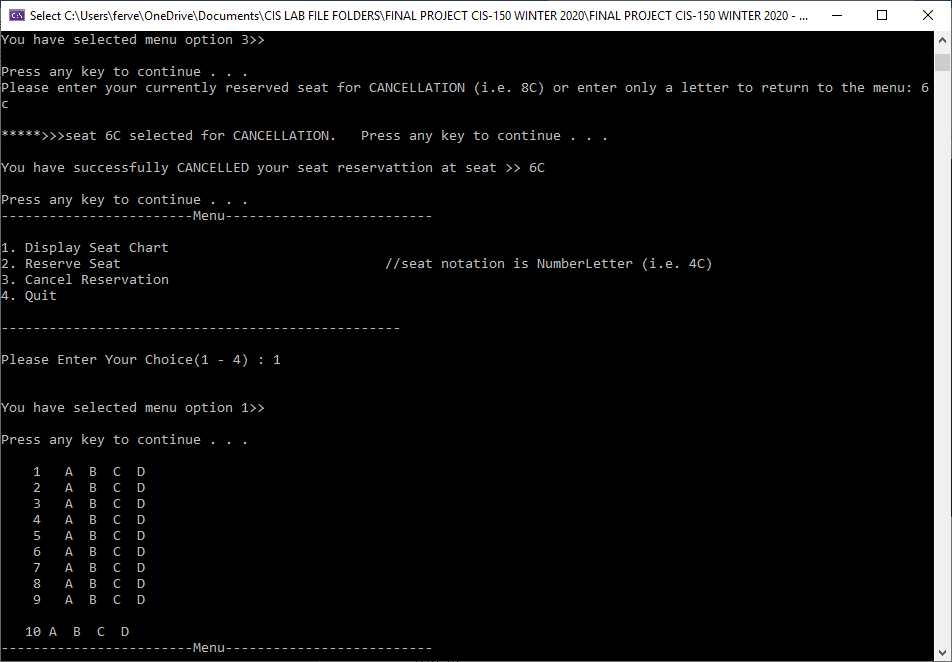
TEST 10



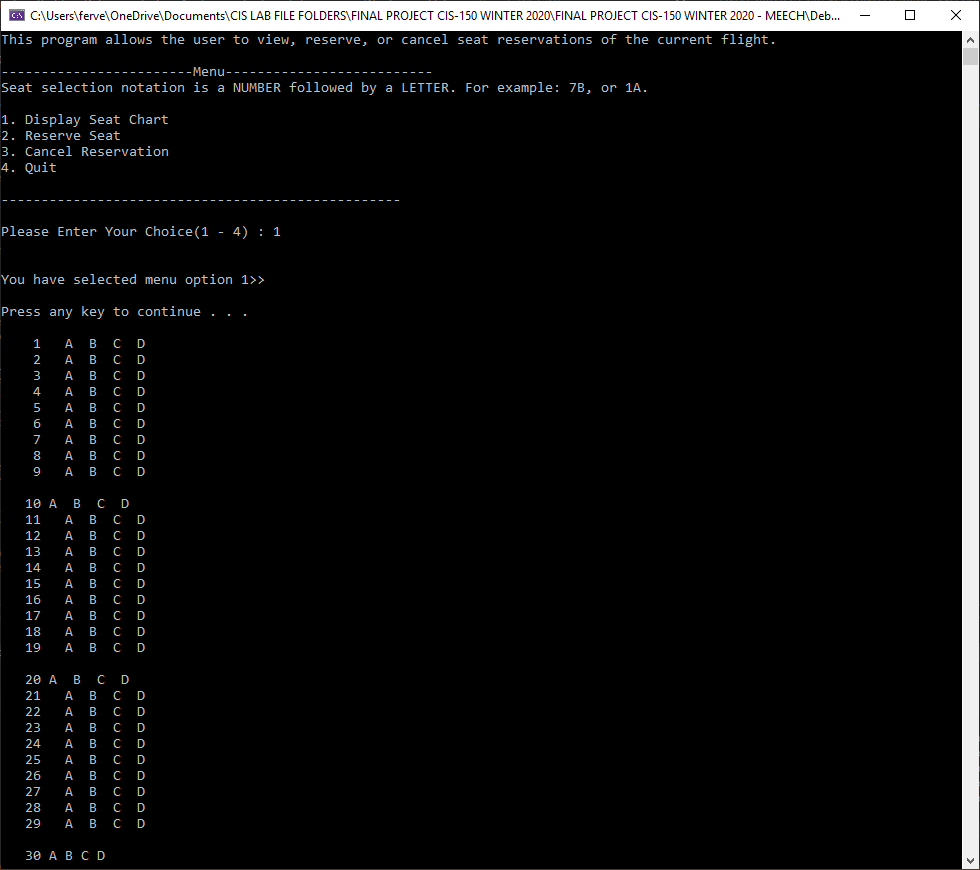
TEST 11



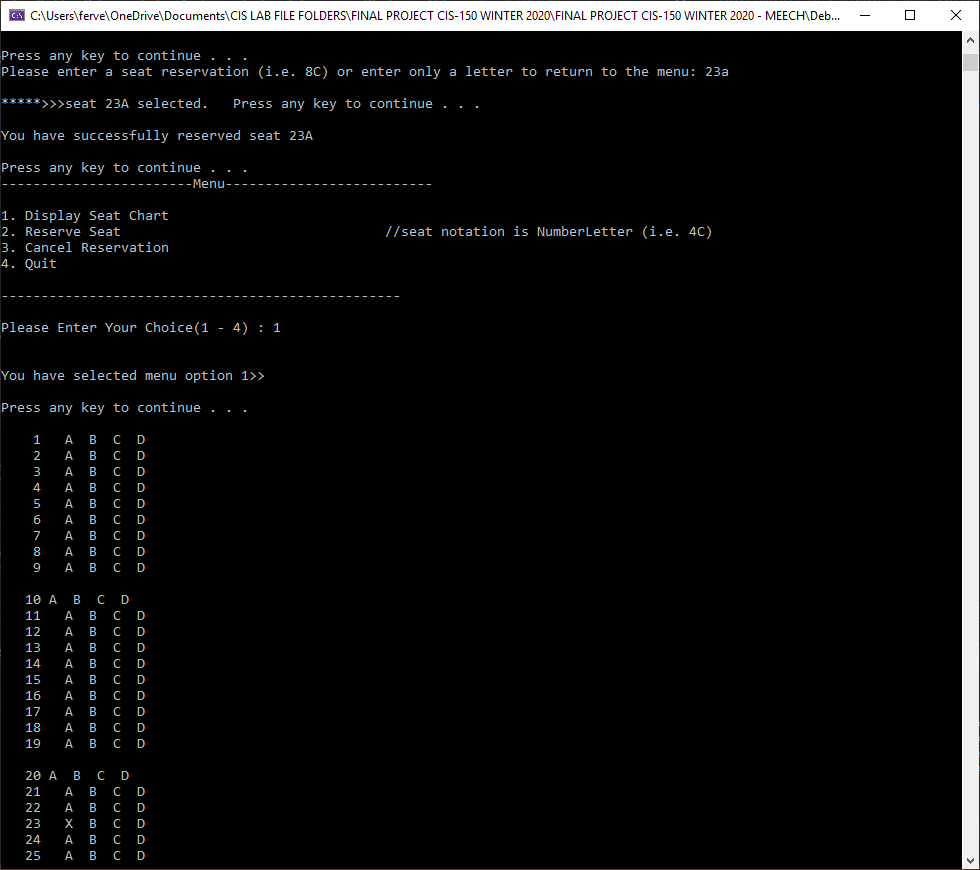
TEST 12



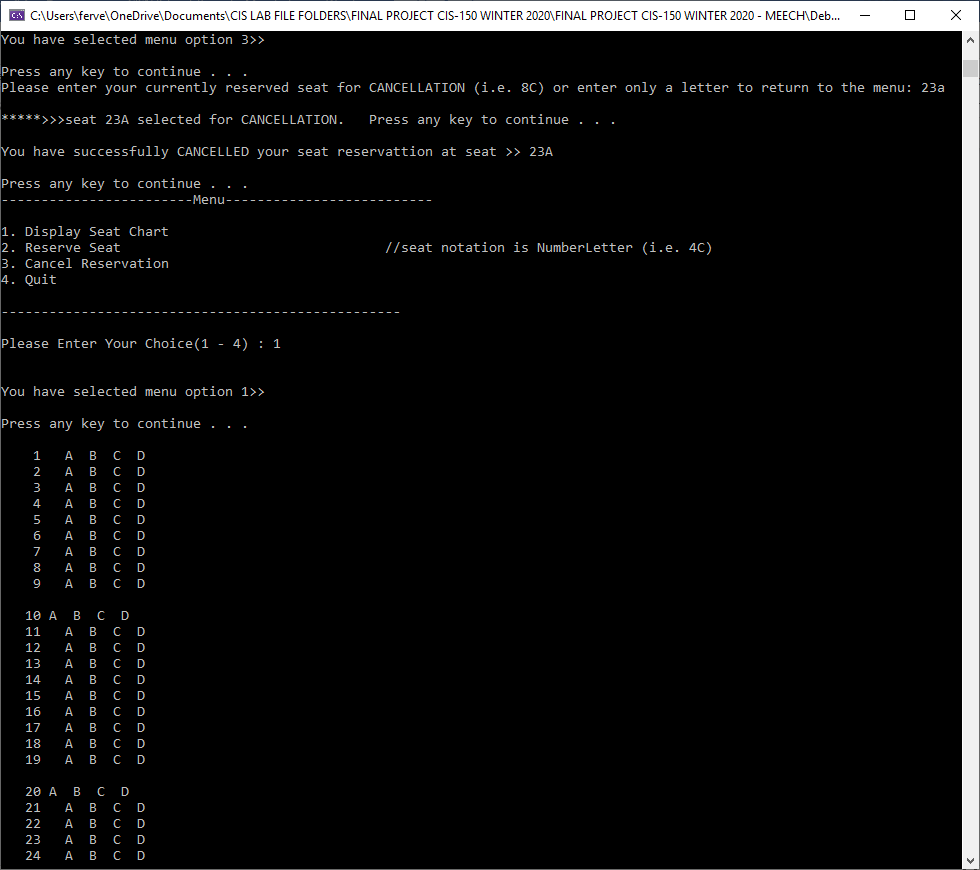
TEST 13



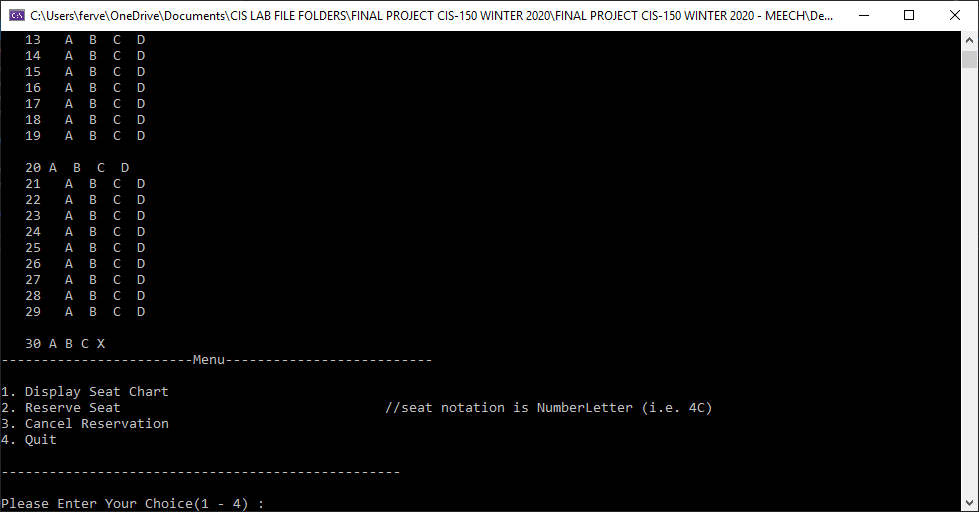
TEST 14

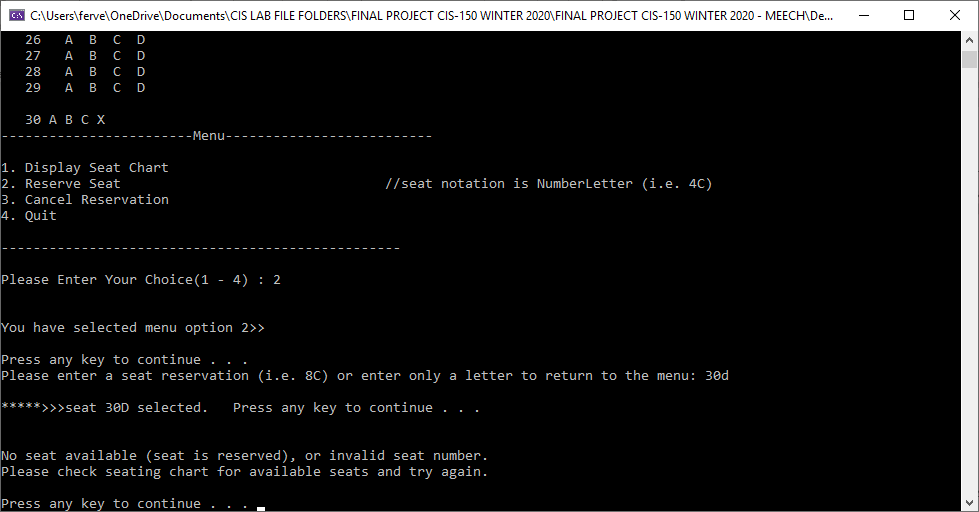


TEST 15

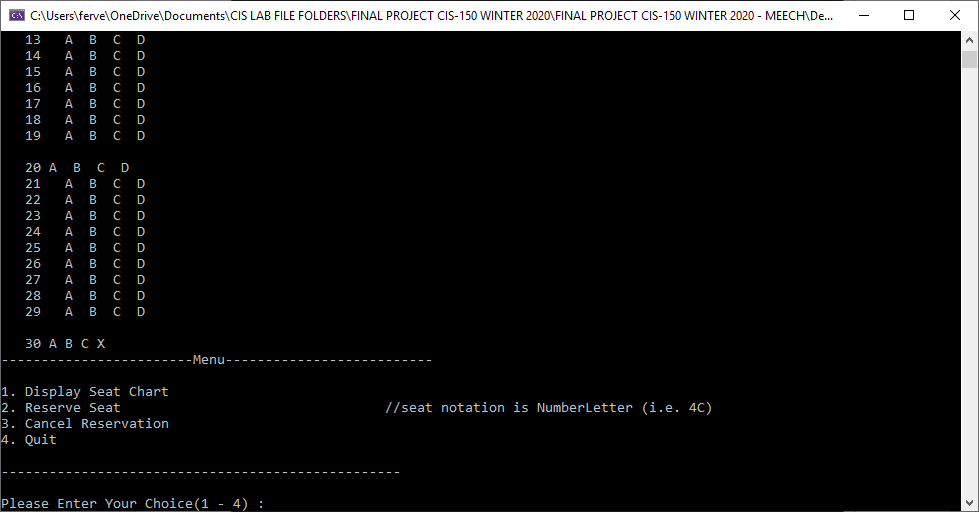


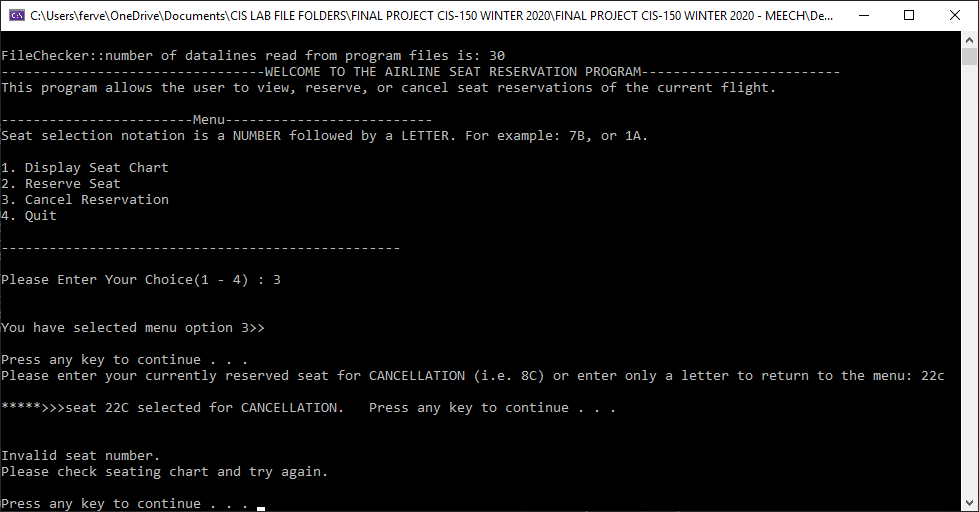
TEST 16





TEST 17





# **Component 4: EXTRA CREDIT – COMPLETED – WILL USE VECTORS AND CLASSES/OBJECTS INSTEAD OF ARRAYS TO ACHIEVE THIS**