**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**cs1713p2Provided.c by your name**

**Purpose:**

**This program reads customer flight reservations to produce output**

**suitable as a Reservation Request.**

**Command Line Arguments:**

**reserve -c customerReservationFileName**

**Input:**

**Stream input file which contains many customer reservations, each**

**containing possibly many flight requests. There are three different**

**kinds of lines of data for each reservation:**

**- Customer Identification Information:**

**o One line per reservation request (separated by spaces)**

**cGender szBirthDt szEmailAddr szFullName**

**1c 10s 40s 30s (may contain spaces)**

**- Customer Address Information**

**o One line per reservation request (separated by commas)**

**szStreetAddress szCity szStateCd szZipCd**

**30s (may contain spaces) 20s 2s 5s**

**- Flight Request:**

**o 0 to many of them per reservation (terminated by END in the Flight ID)**

**szFlightId iNumSeats**

**10s 4d**

**Results:**

**Prints each Reservation in a readable format.**

**Examples:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Flight Reservation Request \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**petem@xyz.net Pete Moss (M 1986/01/01)**

**123 Boggy Lane**

**New Orleans, LA 70112**

**Flight Seats**

**H100.15005 2**

**H222.15005 2**

**H200.15010 2**

**H333.15010 2**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Flight Reservation Request \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**pcorn@abc.net Pop Corn (M 1957/02/02)**

**456 Kernel**

**San Antonio, TX 78210**

**Flight Seats**

**H222.15005 1**

**HXXX.XXXXX 1**

**H333.15010 1**

**Returns:**

**0 normal**

**-1 invalid command line syntax**

**-2 show usage only**

**-3 error during processing, see stderr for more information**

**Notes:**

**reserve -? will provide the usage information. In some shells,**

**you will have to type reserve -\?**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// If compiling using visual studio, tell the compiler not to give its warnings**

**// about the safety of scanf and printf**

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "cs1713p2.h"

Flight findFlight(int count, char\* flightId, int reqSeats, Flight flightM[]);

FILE \*pFileCust; // stream Input for Customer Reservation data

FILE \*pFileFlight;

int main(int argc, char \*argv[])

{

char \*pszCustomerFileName = NULL;

char \*pszFlightFileName = NULL;

Flight flightM[MAX\_FLIGHTS];

int flightCount = 0;

// Process the command switches

processCommandSwitches(argc, argv, &pszCustomerFileName, &pszFlightFileName);

// open the Customer Reservation stream data file

if (pszCustomerFileName == NULL)

exitError(ERR\_MISSING\_SWITCH, "-c");

if (pszFlightFileName == NULL)

exitError(ERR\_MISSING\_SWITCH, "-f");

pFileFlight = fopen(pszFlightFileName, "r");

if(pFileFlight == NULL)

exitError(ERR\_CUSTOMER\_RES\_FILENAME, pszCustomerFileName);

pFileCust = fopen(pszCustomerFileName, "r");

if (pFileCust == NULL)

exitError(ERR\_CUSTOMER\_RES\_FILENAME, pszCustomerFileName);

**// process the Reservations**

printFlights(flightM, flightCount);

processReservations(flightM, flightCount);

printf("Flight Information after processing Reservations\n");

processReservations(flightM, flightCount);

printf("Flight Information after processing Reservations\n");

processReservations(flightM, flightCount);

printf("Flight Information after processing Reservations\n");

processReservations(flightM, flightCount);

printf("Flight Information after Processing Reservations\n");

printFlights(flightM, flightCount);

fclose(pFileCust);

printf("\n"); // included so that you can put a breakpoint on this line

return 0;

}

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* processReservations \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void processReservations()**

**Purpose:**

**Reads and prints three types of data records: identification,**

**address, and flight requests.**

**Parameters:**

**n/a**

**Notes:**

**pFileCust must be open**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

void processReservations(Flight flightM[], int iFlightCount){

char szInputBuffer[100];

int iscanfCnt; // for returning sscanf success

Customer customer; // customer identification and address

FlightRequest flightRequest; // flight request

Flight flightFound;

double totalCost = 0;

**// read customer info until EOF**

while (fgets(szInputBuffer, 100, pFileCust) != NULL)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Flight Reservation Request \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

iscanfCnt = sscanf(szInputBuffer, "%c %10s %40s %30[^\n]\n"

, &customer.cGender

, customer.szBirthDt

, customer.szEmailAddr

, customer.szFullName);

**// Check for bad customer identification data**

if (iscanfCnt < 4)

exitError(ERR\_CUSTOMER\_ID\_DATA, szInputBuffer);

printf("%s %s (%c %s)\n"

, customer.szEmailAddr

, customer.szFullName

, customer.cGender

, customer.szBirthDt);

//read an address record

if (fgets(szInputBuffer, 100, pFileCust) == NULL)

exitError(ERR\_CUSTOMER\_ADDRESS\_DATA, szInputBuffer);

iscanfCnt = sscanf(szInputBuffer, "%30[^,], %20[^,], %2[^,], %5[^\n]\n"

, customer.szStreetAddress

, customer.szCity

, customer.szStateCd

, customer.szZipCd);

**// Check for bad customer address data**

if (iscanfCnt < 4)

exitError(ERR\_CUSTOMER\_ADDRESS\_DATA, szInputBuffer);

printf("%s\n%s %s %s\n"

, customer.szStreetAddress

, customer.szCity

, customer.szStateCd

, customer.szZipCd);

**// Print a heading for the Flight Requests**

printf(" %-10s\t%-5s\t%-10s\t%-4s\n", "Flight", "Seats", "Unit Price", "Cost");

**// There are many flight requests for a customer.**

**// We expect to receive a szFlightId equal to "END"**

**// marking the end of the flight requests.**

**// Read flight requests until EOF.**

while (fgets(szInputBuffer, 100, pFileCust) != NULL)

{

iscanfCnt = sscanf(szInputBuffer, "%10s %4d"

, flightRequest.szFlightId

, &flightRequest.iRequestSeats);

**// Check for bad reservation data**

if (iscanfCnt < 2)

exitError(ERR\_RESERVATION\_DATA, szInputBuffer);

**// If we received an END in flight, we have reached**

**// the end of the requests for this customer**

if (strcmp(flightRequest.szFlightId, "END") == 0)

break;

flightFound = findFlight(iFlightCount, flightRequest.szFlightId, flightRequest.iRequestSeats, flightM);

printf(" %-10s\t%5d\t", flightRequest.szFlightId

, flightRequest.iRequestSeats);

if(flightFound.iAvailSeatCnt == -1)

printf("%-10s\n", "\*\*\* flight ID not found");

else if(flightFound.iAvailSeatCnt == -2)

printf("%-10s\n", "\*\*\* insufficient seats");

else{

totalCost += flightFound.dSeatPrice \* flightRequest.iRequestSeats;

printf("%-10.2f\t%-4.2f\n", flightFound.dSeatPrice,

flightFound.dSeatPrice \* flightRequest.iRequestSeats);

}

}

printf("\t\tTotal Cost\t\t%.2lf\n", totalCost);

totalCost = 0;

} **// end of while loop for reading customer id records**

}

void printFlights(Flight flightM[], int iFlightCount){

int i;

printf("\tFlight\t\tFrom\tDest\tDepart\tAvail\tUnit Price\n");

for(i=0;i<iFlightCount;i++){

printf("\t%s\t%s\t%s\t%s\t%d\t%.2lf\n", flightM[i].szFlightId,

flightM[i].szFrom,

flightM[i].szDest,

flightM[i].szDepartTm,

flightM[i].iAvailSeatCnt,

flightM[i].dSeatPrice);

}

}

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* processCommandSwitches \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void processCommandSwitches(int argc, char \*argv[], char \*\*ppszCustomerFileName)**

**Purpose:**

**Checks the syntax of command line arguments and returns the filenames.**

**If any switches are unknown, it exits with an error.**

**Parameters:**

**I int argc count of command line arguments**

**I char \*argv[] array of command line arguments**

**O char \*\*ppszCustomerFileName customer reservation file name**

**Notes:**

**If a -? switch is passed, the usage is printed and the program exits**

**with USAGE\_ONLY.**

**If a syntax error is encountered (e.g., unknown switch), the program**

**prints a message to stderr and exits with ERR\_COMMAND\_LINE\_SYNTAX.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

void processCommandSwitches(int argc, char \*argv[], char \*\*ppszCustomerFileName,

char \*\*ppszFlightFileName){

int i;

for (i = 1; i < argc; i++)

{

**// check for a switch**

if (argv[i][0] != '-')

exitUsage(i, ERR\_EXPECTED\_SWITCH, argv[i]);

**// determine which switch it is**

switch (argv[i][1])

{

case 'c': // Customer Reservation File Name

if (++i >= argc)

exitUsage(i, ERR\_MISSING\_ARGUMENT, argv[i - 1]);

else

\*ppszCustomerFileName = argv[i];

break;

case 'f':

if(++i >= argc)

exitUsage(i, ERR\_MISSING\_ARGUMENT, argv[i -1]);

else

\*ppszFlightFileName = argv[i];

break;

case '?':

exitUsage(USAGE\_ONLY, "", "");

break;

default:

exitUsage(i, ERR\_EXPECTED\_SWITCH, argv[i]);

}

}

}

int getFlights(Flight flightM[]){

char szInputBuffer[100]; **// input buffer for a line of text**

int iscanfCnt; **// for returning sscanf success**

int flightCnt = 0;

Flight flight;

printf("Original Flight Information\n");

//printf(" Flight Id\tFrom\tDest\tDepart\tAvail\tUnit Price\n");

// read customer info until EOF

while (fgets(szInputBuffer, 100, pFileFlight) != NULL){

iscanfCnt = sscanf(szInputBuffer, "%s %s %s %s %i %lf\n",

flight.szFlightId,

flight.szFrom,

flight.szDest,

flight.szDepartTm,

&flight.iAvailSeatCnt,

&flight.dSeatPrice);

flightCnt++;

if(iscanfCnt < 6)

exitError(ERR\_FLIGHT\_DEF\_DATA, szInputBuffer);

if(flightCnt >= MAX\_FLIGHTS)

exitError(ERR\_TOO\_MANY\_FLIGHTS, szInputBuffer);

flightM[flightCnt-1] = flight;

}

return flightCnt;

}

Flight findFlight(int count, char\* flightId, int reqSeats, Flight flightM[]){

int i;

Flight err;

//printf("\*\*\*%s\*\*\*\n", flightId);

for(i=0;i<count;i++){

if(strcmp(flightId, flightM[i].szFlightId) == 0){

if(flightM[i].iAvailSeatCnt - reqSeats >= 0){

flightM[i].iAvailSeatCnt -= reqSeats;

return flightM[i];

}

else{

err.iAvailSeatCnt = -2;

return err;

}

}

}

err.iAvailSeatCnt = -1;

return err;

}

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* exitError \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void exitError(char \*pszMessage, char \*pszDiagnosticInfo)**

**Purpose:**

**Prints an error message and diagnostic to stderr. Exits with**

**ERROR\_PROCESSING.**

**Parameters:**

**I char \*pszMessage error message to print**

**I char \*pszDiagnosticInfo supplemental diagnostic information**

**Notes:**

**This routine causes the program to exit.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

void exitError(char \*pszMessage, char \*pszDiagnosticInfo)

{

fprintf(stderr, "Error: %s %s\n"

, pszMessage

, pszDiagnosticInfo);

exit(ERROR\_PROCESSING);

}

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* exitUsage \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void exitUsage(int iArg, char \*pszMessage, char \*pszDiagnosticInfo)**

**Purpose:**

**If this is an argument error (iArg >= 0), it prints a formatted message**

**showing which argument was in error, the specified message, and**

**supplemental diagnostic information. It also shows the usage. It exits**

**with ERR\_COMMAND\_LINE\_SYNTAX.**

**If this is just asking for usage (iArg will be -1), the usage is shown.**

**It exits with USAGE\_ONLY.**

**Parameters:**

**I int iArg command argument subscript**

**I char \*pszMessage error message to print**

**I char \*pszDiagnosticInfo supplemental diagnostic information**

**Notes:**

**This routine causes the program to exit.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

void exitUsage(int iArg, char \*pszMessage, char \*pszDiagnosticInfo)

{

if (iArg >= 0)

fprintf(stderr, "Error: bad argument #%d. %s %s\n"

, iArg

, pszMessage

, pszDiagnosticInfo);

fprintf(stderr, "reserve -c customerReservationFile\n");

if (iArg >= 0)

exit(-1);

else

exit(-2);

}