Question 1

Function buy tickets ( class of ticket )

Ask about full fare

Ask about concession

Ask about baby tickets

Add all those orders together

Subtract that number from the remaining tickets

Return to menu

Function ask how many tickets are left

Print the following:

Class A VALUE

Class B VALUE

Class C VALUE

Return to menu

Menu

Buy class A tickets

Buy class b tickets

Buy class c tickets

Check how many tickets are left

Exit

Code:

#include <stdio.h>

#include <stdbool.h>

/\* Main function \*/

int main(void) {

/\* Constants \*/

/\* Pricing \*/

const int babyPrice = 0;

/\* Class A \*/

const int classATicketPriceFull = 25;

const int classATicketPriceConc = 15;

/\* Class B \*/

const int classBTicketPriceFull = 15;

const int classBTicketPriceConc = 10;

/\* Class C \*/

const int classCTicketPriceFull = 9;

const int classCTicketPriceConc = 5;

/\* Array to store all pricing. \*/

/\* First [] bracket is class, seccond is for concession status. \*/

const int ticketPricing[3][2] = {

{classATicketPriceFull, classATicketPriceConc},

{classBTicketPriceFull, classBTicketPriceConc},

{classCTicketPriceFull, classBTicketPriceConc}

};

/\* Children \*/

const int babyAgeThreshold=3;

const int childAgeThreshold=12;

/\* Initial ticket availability \*/

const int classATickets=15;

const int classBTickets=60;

const int classCTickets=90;

/\* Variables \*/

/\* Array for all tickets. \*/

int tickets[3] = {classATickets, classBTickets, classCTickets};

/\* Functions \*/

/\* Check how many tickets there are left. \*/

void currentAvailability(void) {

printf("Current Availability:\n"

"Class A: %d\n"

"Class B: %d\n"

"Class C: %d\n"

"\n\n", tickets[0], tickets[1], tickets[2]);

}

/\* Check types of tickets. \*/

void buyTickets(int ticketClass) {

/\* Arguments \*/

/\* int ticketClass : Used for determining which class of ticket the user

\* has selected in the previous menu. \*/

/\* Constants \*/

/\* Variables \*/

int userInput = 0;

int fullTickets = 0;

int concTickets = 0;

int babyTickets = 0;

int orderTicketCount = 0;

int ticketPriceTotal = 0;

/\* Testing purposes only: \*

printf(

"Class A tickets left: %d\n"

"Class B tickets left: %d\n"

"Class C tickets left: %d\n",

tickets[0],

tickets[1],

tickets[2]);

/\*\*/

/\* Ask about full fare tickets. \*/

printf("How many full fare tickets? ");

scanf("%d",&userInput);

orderTicketCount = orderTicketCount + userInput;

fullTickets = userInput;

ticketPriceTotal = userInput \* ticketPricing[ticketClass][0];

/\* Ask about concession tickets. \*/

printf("How many concession tickets? "

"(Seniors and children over %d and under %d.) ",

babyAgeThreshold, childAgeThreshold);

scanf("%d",&userInput);

orderTicketCount = orderTicketCount + userInput;

concTickets = userInput;

ticketPriceTotal = ticketPriceTotal +

userInput \* ticketPricing[ticketClass][1];

/\* Ask about child tickets. \*/

printf("How many children under the age of %d? ", babyAgeThreshold);

scanf("%d",&userInput);

printf("\n");

orderTicketCount = orderTicketCount + userInput;

babyTickets = userInput;

/\* Testing purposes only \*

printf("orderTicketCount == %d\n"

"tickets[ticketClass] == %d\n",

orderTicketCount,

tickets[ticketClass] );

/\*\*/

/\* Check if the customer has tried to order more tickets than are

\* available. \*/

if ( orderTicketCount <= tickets[ticketClass] ) {

printf("You have ordered %d full cost ticket(s), "

"%d concession/child ticket(s) and %d baby tickets.\n",

fullTickets, concTickets, babyTickets);

printf("Total ticket price comes to: $%d\n", ticketPriceTotal);

tickets[ticketClass] = tickets[ticketClass] - orderTicketCount;

} else {

/\* We'll tell the user how many tickets are left here just so they

\* don't have to go and check for themselves. \*/

printf("Tickets ordered exceed number of available tickets.\n");

currentAvailability();

}

}

bool contloop = true;

while ( contloop == true ) {

/\* Body \*/

/\* Main menu text \*/

printf("\n\nTrain Ticketing System\n\n"

"1 - Class A\n"

"2 - Class B\n"

"3 - Class C\n"

"4 - Current Availability\n"

"5 - Exit\n"

"Enter your option...\n\n");

/\* Handle menu selection \*/

int option = 0;

scanf("%d", &option);

printf("\n");

switch (option) {

case 1: printf("Class A\n");

buyTickets(0);

break;

case 2: printf("Class B\n");

buyTickets(1);

break;

case 3: printf("Class C\n");

buyTickets(2);

break;

case 4: currentAvailability();

break;

default:printf("Exiting...\n\n");

contloop = false;

break;

}

}

/\* I have put a newline here so that my command prompt doens't get spat out

\* on the end of the last line of text. \*/

printf("\n\n");

return 0;

}

Screenshots:

A close up of a black background

Description automatically generated

Question 2

Loop until user answers -1

Generate two random numbers x and y.

Loop until correct.

Print: How much is x + y? Type -1 to exit.

Check if correct or matches -1.

Give appropriate response at random

Code:

#include <stdio.h>

#include <stdbool.h>

#include <time.h>

#include <stdlib.h>

int main(void) {

srandom(time(NULL));

/\* Variables \*/

int r1 = random() % 5;

int r2 = random() % 5;

/\* This is needed to maintain the loop till the user opts out. \*/

bool cont = true;

/\* This is the main game loop. \*/

void gameLoop(void) {

/\* Variables \*/

int answer = 0;

/\* Create two new numbers. \*/

r1 = random() % 10;

r2 = random() % 10;

printf("Add the following numbers.\n\n");

while ( answer != r1 + r2 ) {

/\* Ask the user what the answer is. \*/

printf("How much is %d + %d? Enter -1 to quit. ", r1, r2);

scanf("%d", &answer);

printf("\n");

/\* Handle user response. \*/

int rresp = random() % 4 + 1;

if ( answer == -1 ) {

/\* Quit out of the main game loop. \*/

cont = false;

/\* Make the answer correct so that the user doesn't get asked

\* again when they're trying to leave the game. \*/

answer = r1 + r2;

printf("Exiting.\n\n");

} else {

/\* Handle responses. \*/

if ( answer == r1 + r2 ) {

switch ( rresp ) {

case 1: printf("Very good!");

break;

case 2: printf("Excellent!");

break;

case 3: printf("Nice work!");

break;

case 4: printf("Keep up the good work!");

break;

}

} else {

switch ( rresp ) {

case 1: printf("No. Please try again.");

break;

case 2: printf("Wrong. Try once more.");

break;

case 3: printf("Don't give up!");

break;

case 4: printf("No. Keep trying.");

break;

}

}

}

/\* Keeps the questions spaced out nicely and stops the terminal

\* prompt from starting on the same line when you choose to quit. \*/

printf("\n\n");

}

}

while ( cont == true ) {

gameLoop();

}

return 0;

}

Screenshots

A screenshot of a cell phone

Description automatically generated

Question 3

2D array for storing seat status with

Function current availability

Print: map of seats in a grid marked as occupied or empty with aisle letters and seat numbers.

Function returning int of available seats

For x

For y

If seat is empty

Empty++

Return available seats

Function mark seats as reserved ( amount to reserve )

For x or reserved

For y or reserved

If seat is empty

fill it then notify the user of the seat letter and number

reserved++

reset y counter to start in the next row

Function handle reserving tickets ( amount, xpos, ypos )

Set starting position to x and y if they were specified

For x or amount

For y amount

If seat is empty

If this is the first empty seat in a row that we have found

Mark this position as the index

Seatsfound++

If amount == seats found

Mark this position as the tail

Else if seat isn’t empty

Reset seats found counter

Reset y counter to start the next row

If the correct amount of seats were found

Tell the user the cost and location of the seats

Use available seats map function to display

Ask user if they would like to reserve the seats

If yes

Mark the reservation using function

If no

Return to menu

Else if not enough seats found

Offer to dump seats in different rows

Yes reserve / no return to menu

Function input for reservation

Ask how many seats

Check if this number exceeds 10 or the number of remaining seats in the cinema

Ask if computer may allocate for user

If yes

Allocate automatically using function

If no

Ask user what row and column

Allocate using function but pass row / col numbers

Function menu

Display a menu linking to the major functions:

Reserve tickets

Show current availability

Count available seats

Exit

Main

Loop the menu every time the major functions exit unless the menu exit function is used

Code:

/\* Input output \*/

#include <stdio.h>

/\* atoi \*/

#include <stdlib.h>

/\* true/false statements \*/

#include <stdbool.h>

/\* gdb debugging \*/

#include <signal.h>

/\* string handling, might not need this. \*/

#include <string.h>

int main(void) {

int ticketCost = 15;

/\*int seats[10][8] = {0};

/\*\*/

/\* Define custom initial array. (testing purposes only) \*/

int seats[10][8] = {

{1, 0, 1, 0, 0, 1, 0, 0},

{0, 1, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 0, 0, 0, 1, 0},

{0, 1, 0, 1, 0, 1, 0, 1},

{0, 0, 1, 0, 0, 1, 0, 0},

{1, 0, 0, 1, 0, 0, 1, 0},

{1, 0, 0, 1, 0, 0, 1, 0},

{1, 0, 0, 1, 0, 0, 1, 0},

{1, 0, 0, 1, 0, 0, 1, 0},

{1, 0, 0, 1, 0, 0, 1, 0}

};

/\*\*/

/\* Clear junk out of stdin to stop it causing problems reading input. \*/

void clearstdin(void) {

/\* Clear junk out of stdin. \*

int c;

while ((c = getchar()) != '\n' && c != EOF) { }

/\*\*/

fflush(stdin);

}

/\* Print a map of the available seats \*/

void currentAvailability( int xstart, int ystart, int ammount ) {

int x = 0;

int y = 0;

char aisleLetter;

printf(" 1 2 3 4 5 6 7 8 9 10\n");

for ( y; y < 8; y++ ) {

/\* This will convert the aisle number to a letter. \*/

aisleLetter = y + 'A';

printf("%c ", aisleLetter);

for ( x; x < 10 ; x++ ) {

if (ammount != 0 &&

x >= xstart &&

x <= xstart + ammount - 1 &&

y == ystart ) {

printf("? ");

} else {

if ( seats[x][y] == 0 ) {

printf(". ");

} else {

printf("R ");

}

}

}

printf("\n");

x = 0;

}

}

/\* Print the number of available seats left \*/

int availableSeats(void) {

int freeCount = 0;

int x = 0;

int y = 0;

for ( y; y < 8; y++ ) {

for ( x; x < 10; x++ ) {

if ( seats[x][y] == 0 ) {freeCount++;}

}

x = 0;

}

return freeCount;

}

/\* Reserve available seats \*/

/\* This function will actually accept more than 10 seats at a time,

\* the restriction for 10 or less seats is placed on the function call. \*/

void reserveTickets(int orderSize, int row , int seatno) {

int x = 0;

int indexSeatX = 0;

int tailSeatX = 0;

int y = 0;

int indexSeatY = 0;

int tailSeatY = 0;

int contiguousSeats = 0;

/\* Reserve a group of seats in a row. \*

void markReservation( int xstart, int ystart, int reservation) {

/\* Reset and reuse x and y for iterating through the seats because I

\* can't be bothered typing "start". \*

x = xstart;

y = ystart;

int reserved = 0;

/\* Iterate through the seats and mark the correct ammount as

\* reserved. \*

for ( y; y < 8 && reservation != reserved; y++ ) {

for ( x; x < 10 && reservation != reserved; x++) {

seats[x][y] = 1;

/\* Testing purposes only. \*

printf("Checking seats: x%d y%d\n", x, y);

/\*\*

reserved++;

/\* Testing purposes only. \*

printf("%d\n", reserved);

/\*\*

}

x = 0;

}

}

\*/

/\* Allocate seats given that they're not taken. \*/

void markReservation( int xstart, int ystart, int reservation ) {

/\* Reset and reuse x and y for iterating through the seats because I

\* can't be bothered typing "start". \*/

x = xstart;

y = ystart;

int reserved = 0;

/\* Iterate through the seats and mark the correct ammount as

\* reserved. \*/

for ( y; y < 8 && reservation != reserved; y++ ) {

for ( x; x < 10 && reservation != reserved; x++) {

/\* Testing purposes only. \*

printf("Checking seats: x%d y%d\n", x, y);

/\*\*/

if ( seats[x][y] == 0 ) {

seats[x][y] = 1;

char sletter = y + 'A';

char snumber = x + 'A';

printf( "You have been placed into seat %c%d.\n",

sletter, snumber );

reserved++;

/\* Testing purposes only. \*

printf("%d\n", reserved);

/\*\*/

}

}

x = 0;

}

}

/\* For testing purposes only. \*

printf("Running reserve tickets function now.\n");

/\*\*/

/\* Testing purposes only. \*

printf("row:%d\n", row);

printf("seatno:%d\n", seatno);

/\*\*/

/\* Only set the row if the user specified something,

\* otherwise just start at the start.

\* This has to check if it's already zero so we don't start at -1. \*/

if ( row != 0 ) { y = row - 1; }

if ( seatno != 0 ) { x = seatno - 1; }

/\* Testing purposes only. \*

printf("x:%d\n", x);

printf("y:%d\n", y);

/\*\*/

/\* Testing purposes only. \*

printf("orderSize:%d\n", orderSize);

printf("contiguousSeats:%d\n\n", contiguousSeats);

/\*\*/

for ( y;

y < 8 && contiguousSeats != orderSize;

y++ ) {

/\* Restrict the allocation to one row so we don't spill across to

\* the next row. \*/

contiguousSeats = 0;

/\* Move through the columns (across the aisle) starting with 0

\* (seat 1) \*/

for ( x;

x < 10 && contiguousSeats != orderSize;

x++ ) {

/\* Testing purposes only. \*

printf("Checking seats: x%d y%d\n", x, y);

/\*\*/

/\* For testing purposes only \*

printf("Reserved: %d\n\n", seats[x][y]);

/\*\*/

if ( seats[x][y] == 0 ) {

/\* Set the starting position of the chain of empty

\* seats if this is the first one we have found. \*/

if ( contiguousSeats == 0 ) {

indexSeatX = x;

indexSeatY = y;

}

/\* The seat is empty so we count it and move on. \*/

contiguousSeats = contiguousSeats + 1;

/\* For testing purposes only. \*

printf("Contiguous seats: %d\n\n", contiguousSeats);

/\*\*/

/\* Make sure we track where the chain of empty seats ends. \*/

if ( contiguousSeats == orderSize ) {

tailSeatX = x;

tailSeatY = y;

}

} else {

contiguousSeats = 0;

}

}

/\* Start at the start of the next row. \*/

x = 0;

}

/\* For testing purposes only. \*

printf( "Head:x%dy%d\n"

"Tail:x%dy%d\n",

indexSeatX,

indexSeatY,

tailSeatX,

tailSeatY);

/\*\*/

/\* Increment the ASCII value by A (41) in order to make aisle letters

\* out of the aisle number. \*/

char indexSeatLetter;

indexSeatLetter = indexSeatY + 'A';

char tailSeatLetter;

tailSeatLetter = tailSeatY + 'A';

/\* For testing purposes only. \*

printf("indexSeatLetter:%c\n", indexSeatLetter);

printf("tailSeatLetter:%c\n", tailSeatLetter);

/\*\*/

/\* We have found a group of seats or there is no group of seats

\* large enough to accomodate the order. \*/

char yesno[10];

if ( contiguousSeats == orderSize ) {

/\* Place order successfuly. \*/

int price = ticketCost \* orderSize;

printf("Seats %c%d through %c%d are free, would you like to "

"reserve them?\n"

"The price will be $%d.\n\n",

indexSeatLetter,

indexSeatX + 1,

tailSeatLetter,

tailSeatX + 1,

price);

currentAvailability( indexSeatX, indexSeatY, orderSize );

/\* Take input. \*/

clearstdin();

fgets(yesno, 8, stdin);

/\* Handle the user deciding where to be seated. \*/

if ( strncmp(yesno, "y", 1) == 0 ) {

printf("Seats reserved.\n");

markReservation(indexSeatX, indexSeatY, orderSize);

} else {

printf("Seat reservation cancelled.");

}

} else {

/\* Can't find enough contiguous seats.

\* Ask if they want to get dumped anywhere. \*/

printf( "Not enough seats in one row to satisfy request.\n"

"Would you like to be placed into the remaining available "

"seats?\n");

clearstdin();

fgets(yesno, 8, stdin);

/\* Handle the user deciding where to be seated. \*/

if ( strncmp(yesno, "y", 1) == 0 ) {

markReservation( 0, 0, orderSize );

}

}

}

/\* Ask the user how they would like their seats selected. \*/

void reservationInput(void) {

printf("Reserve tickets:\n");

/\* Variables \*/

int selection;

int seatAmmount = 0;

int seatRow = 0;

int seatNumber = 0;

/\* Used to store strings the user inputs. \*/

unsigned char selectionChar[10] = {'a'};

/\* Debugging interupt. \*

raise(SIGINT);

/\*\*/

/\* How many? \*/

printf("How many seats would you like to reserve? ");

/\*scanf("%d [1-10]", &selection);\*/

/\*clearstdin();\*/

fgets(selectionChar, 10, stdin);

seatAmmount = atoi(selectionChar);

/\* For testing purposes only. \*

printf("%d\n", seatAmmount);

/\*\*/

/\* Is the machine deciding where they are for you? Y/N \*/

printf("Would you like the computer to allocate seats for "

"you? (y/n) ");

/\*scanf("%c", &selectionChar);\*/

clearstdin();

fgets(selectionChar, 10, stdin);

/\* Handle the user deciding where to be seated. \*/

if ( strncmp(selectionChar, "n", 1) == 0 ) {

printf("What row & seat would you like to reserve the "

"seats from?\n"

"Row (A-H): ");

/\*scanf("%c [a-h]", &selectionChar[1]);\*/

clearstdin();

fgets(selectionChar, 10, stdin);

/\* Testing purposes only. \*

printf("selectionChar[1]:%d\n", selectionChar[1]);

printf("(int)selectionChar[1]:%d\n", (int)selectionChar[1]);

/\*\*/

/\* Subtract the ascii value for "a" to create a useful number.

\* I'm assuming the input is lowercase but I could convert the

\* string to all uppers then assume uppercase to avoid issues. \*/

seatRow = selectionChar[0] - 'a' + 1;

/\* Testing purposes only. \*

printf("%d\n", seatRow);

/\*\*/

/\* Ask about seat number. \*/

printf("Seat number (1-10): ");

/\*scanf("%d [1-10]", &seatNumber);\*/

clearstdin();

fgets(selectionChar, 10, stdin);

seatNumber = atoi(selectionChar);

/\* For testing purposes only. \*

printf("%d", seatNumber);

/\*\*/

}

/\* For testing purposes only. \*

printf("%d\n", seatAmmount);

/\*\*/

/\* Only accept orders of 10 or less tickets at a time. \*/

if ( seatAmmount > 10 ) {

/\* Invalid order size that is greater than 10. \*/

printf("Sorry, a maximum of 10 tickets can be reserved "

"at one time.");

} else if ( seatAmmount > availableSeats() ) {

/\* Not enough seats left. \*/

printf( "Sorry, you have requested more seats than the remaining "

"quantity, there are only %d seats available.",

availableSeats() );

} else {

/\* Reserve the tickets. \*/

reserveTickets(seatAmmount, seatRow, seatNumber);

/\*\*/

}

}

/\* Make a bool to decide whether to continue looping the menu or quit. \*/

bool cont = true;

int menu(void) {

/\* Main menu text \*/

printf("\n\nCinema Ticketing System\n\n"

"1 - Reserve Tickets\n"

"2 - Show Current Availability\n"

"3 - Count Available Seats\n"

"4 - Exit\n"

"Enter your choice...\n\n");

/\* Handle menu selection \*/

/\*clearstdin();\*/

char option[10];

/\*scanf("%d", &option);\*/

/\*clearstdin();\*/

fgets(option, 10, stdin);

int optionInt = atoi(option);

printf("\n");

/\* For testing purposes only. \*

printf("%d\n", test);

/\*\*/

switch (optionInt) {

case 1: reservationInput();

break;

case 2: printf("Current availability:\n");

currentAvailability(0, 0, 0);

break;

case 3: printf("Available seat count:\n"

"%d", availableSeats());

break;

default:printf("Exiting...\n\n");

cont = false;

break;

}

}

while ( cont == true ) {

menu();

}

return 0;

}

Screenshots:A screenshot of a cell phone

Description automatically generatedA close up of a computer

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA picture containing laptop

Description automatically generatedA picture containing black, laptop

Description automatically generated