Developer

1. Keh Yi Qian
2. Lai Chin Hin
3. Neo Ze Zhen

Task Division

	Name	Modules	Description	*A2 Contribution (Overall, %)	
1.	Keh Yi Qian	View	Allow the users to see the synopsis of different movies.	1	3
		Booking	Display the available and not availble seat. Allow the users to book the movie tickets that their like and available to choose the sear based on their favourite. And store the booking details into the cart.	2	10
		Delete	Allow the users to delete the movie's ticket in the cart	3	10
		Payment	Allow the users to purchase the movie's tickets and pay with different methods. Also, will find change to the customers when customers use cash to pay.	4	10
		Discount	Provided the discount when customers purchase in the range of number of tickets and provided member discount	5	10
		Receipt (jointly)	Store the payment details of every transaction in a receipt. Also, save every receipt into a text file and available to display the receipt in the payment part.	6	5
2.	Lai Chin Hin	View	Allow users to view their membership details when they log in.	1	3
		Register	Allow users to register their membership with different levels of membership.	2	10
		Update	Allow users to update their membership details, such as name, IC, etc.	3	10
		Save	Updated membership details will be recorded in a text file.	4	5
		Authorization	Users can only login to their own account by inputting the correct user ID and password.	5	5
3.	Neo Ze Zheng	View	Allow users to see the history record.	1	4

		Search	Allow users to search the transaction id and display the receipt.	2	10
		Receipt (jointly)	Store the payment details of every transaction in a receipt. Also, save every receipt into a text file and available to display the receipt in the payment part.	3	5

* Depends on the evaluation of the markers as well;

Objective

- Create a Cinema Ticketing System to sell movie tickets and also reduce worker workload.

    - Permit customers to select the movie summary they want to read.

    - Allow customers to purchase the movie tickets that their like and available to choose the sear based on their favourite.

    - Allow customers to delete movie tickets.

    - Allow customers to purchase tickets with different payment methods.

    - Enable customers to sign up for their own memberships that comes with various level.

Pseudocode

Int main()

```
Declare option as an integer and enter as a boolean equal true;
While enter equal true
        Call system_interface();
        Get user's input and store it as option;
        Clear other remaining input;
        Switch on option
        Case1:
                Call option1();
                set enter equal true;
                Break;
        Case2:
                Call payment() and set enter as return value;
                Break;
        Case3:
                Call history() and set enter as the return value;
                Break;
        Case4:
                Call option4() and set enter as the return value;
                Break;
        Case5:
                Call exit_page() and set enter as the return value;
                Break;
        Default:
                Output invalid input. Key in 1-6;
                set enter as true;
pause the program;
return 0;
Terminate;
```

Function: void option1()

```
Declare option1 as boolean and equally true; choice, choice1, count as integer and count equal 0; yes_no
as character;  line, booking1 as string;
As long as the option1 is true
        Call function print1();
        Ask the user to input choice;
        Clear the input buffer
        Switch case based on choice:
        Case 1:
                 Declare bool a = true;
                As long as a is true;
                        Call function print2();
                        Call function array_movie(movies, "Movie_information.txt");
                        Display movie details;
                        Call function print3();
```

Ask the user to input choice1;
Clear the input buffer;
Switch case based on choice1:
Case 1:
Ask the user to input yes_no;
If yes_no is 'Y' or 'y'
call function synopsis();
Otherwise
clear the input buffer and Declare a = true;
Break switch case;
Case 2:
Call function booking();
Declare a = true and option1 = true;
Break switch case;
Case 3:
Call function exit_page() and Declare a = return value;
Default:
Display an error message;
Declare a = true;
Break switch case;
Case 2:
Clear the console screen;
Declare bool b = true;
As long as b is true
Call function cart("cart.txt");
Call function edit_cart() and b = return value;
Break switch case
Case 3:
Declare option1 = return value of function exit_page();
Break switch case;
Default:
Display an error message;
Break switch case
Terminate;

Function: bool exit_page()
Declare exit as character;
Display question for asking user want to exit the page or not;
Ask the user to input exit;
If the uppercase of exit is 'Y'
Return false
Otherwise
Return true;
Terminate;

Function: int array_movie(Movie movies[], const string& filename)
Declare num_movie as integer and equally 0;

Open the movie_file using the given filename;
If the file is successfully opened
    Declare string line;
    As long as there are still lines to read from the movie_file and num_movie is less than max_movie
        Create a new instance of the struct Movie called movie;
        Read the current line from the file and assign its substrings to the corresponding fields of movie;
        Assign the current movie struct to the movies array;
        Nim_movie increase 1;
Otherwise
    Display an error message;
Close the movie file;
Return the value of num_movie;

Function: void synopsis()

Declare line, synopsis_line, movie_ID as string; count as integer; continues as character; a as boolean and set a to true;
As long as a is true
    Ask the user to input the movie id that they want to see the synopsis;
    If the format of the movie id is incorrect
        Display an error message;
        Set a to true;
    Otherwise
        open the moviefile using "Movie_information.txt";
        If the file is open
            Declare found_movie_id as boolean and set it to false;
            Start a for loop with i from 0 to num_movie and plus 1 every time
                Read each line in the file store it in line;
                If the movie id is found in the moviefile and not found '.' in the movie id
                    Declare filename as string and equally movie id + '.txt';
                    Open the synopsis_file using the given filename;
                    As long as there still can read line from synopsis_file and store it into synopsis_line
                        Display synopsis_line;
                    Set found_movie_id to true;
                    Break to escape the for loop;
            If not found_movie_id
                Display an error message;
            Close the moviefile;
            Asking the user need to continue the while loop or not;
Terminate;

Function: void booking()

Declare a and find as Boolean and set a to true while set b to false; movie_id, line and seat_code as string; bookcode as the character array with 3 elements
Start a for loop with i from 0 to less than num_movie and plus 1 every time

Start a for loop with j from 0 to less than row and plus 1 every time
Start a for loop with k from 0 to less than column and plus 1 every time
set all seats to '#' for all movies;
Open the infile using "booking.txt";
If infile is open
As long as there still can read seat_code from infile
Mark the corresponding seat that read from infile as 'X' in the 'seats' array;
Close the infile;
As long as a is true
Ask the user to input movie_id;
Clean the input buffer;
If movie_id is Q
Set a to false;
Otherwise, if the format of movie_id is incorrect
Display an error message and set a to true;
Otherwise
Start a for loop with i from 0 to less than num_movie and plus 1 every time;
If movie_id is found in the movie_ID in the movie struct that in the accordingly movies array
Start a for loop with j from 0 to less than 3 and plus 1 every time;
Store the movie_id one by one into ‘bookcode’ array;
Declare number_of_id as integer and which is equally to the last two digits of the movie_id using the programming skill ;
Call seat() function with argument number_of_id;
Call valid_of_seat() function with arguments number_of_id and movie_id;
Set find to true;
Declare continues as character; c as Boolean and equally true;
As long as c is true
Ask the user need to continue to book the ticket or not, if not escape this loop;
If not find
Display an error message;
Terminate;

Function: void seat(int num_of_id)
Display decoration of seat interface;
Display 01 to 10;
Start a for loop with i from 0 to less than row and plus 1 every time;
Display A + i;
Start a for loop with j from 0 to less than column and plus 1 every time;
Display seat status (available or taken) based on the given num_of_id parameter.
Terminate;

Function: void valid_of_seat(int number_of_id, string movie_id)
Declare choose_column and movie_id1 as string and set movie_id1 to movie_id that is already given;
choose_row as character;

Ask the user to input choose_row and choose_column;
Convert choose_row to uppercase;
Error handling for input;
If format of the choose_row and choose_column is true
    Open the outfille using "booking.txt" and let it in append mode;
    Declare choose_column_chart as character array in 2 elements;
    Start a for loop with i from 0 to less than 2 and plus 1 every time;
        Store the choose_column one by one into ‘choose_column_chart’ array;
    Declare choose_column_int as integer and which is convert choose_column into integer using the value in the ‘choose_column_chart’ array and programming skill ;
    If choose_column_char is less than and equal to 10
        If the value in the ‘seats’ array is ‘X’
            Display the message that seat is already booked;
        Otherwise, if the value in the ‘seats’ array is ‘#’
            Set the value in the ‘seats’ array to ‘X’;
            Display the booking successfully message;
            Store number_of_id, choose_row and choose_column into outfile;
            Close the outfile;
            Declare column_row as string and which is the Concatenate of choose_row and choose_column;
            Call the save_into_cart() function with arguments movie_id1, "cart.txt", and column_row;
            Call the seat() function with argument number_of_id;
    Otherwise
        Display an error message for invalid choose_column;
Otherwise
    Display an error message;
Terminate;

int array_cart(Cart carts[])
declare line_in_cartfile as integer and equal 0;
open “cart.txt” as cartfile;
if cartfile is open:
    declare line as string;
    while getting each line from cartfile using getline and line_in_cartfile smaller than max_line_in_cart:
        create a new instance of the struct Cart named cart;
        read the current line and assign its substrings to the corresponding fields of cart;
        Assign current cart to cart array;
        Line_in_cartfile increase 1;
Close the cartfile;
Return line_in_cartfile;
Terminate.

Void save_into_cart(string movie_id1, string filename, string column_row)
Call array_movie(movies, string filename);
Declare name as a string, and price_movie as double, line_in_cart as integer;

```
line_in_cart equal array_cart(carts);
start a for loop with i from 0 to num_movie and plus 1 every time
        if movies[i].movie_ID contains movie_id1
                declare name equal movies[i].movie name;
                declare price_movie equal movies[i].price;
                start a loop for j from 0 to num_movie and plus 1 every time
                        if carts[j].record seat doesn't contain column_row or cart[j].record_id does not
                        contain movie_id1
                                open a filename as writefile in append mode;
                                if writefile is open
                                        output movie_id1, name, column_row, and price_movie from
                                        writefile;
                                close the writefile;
                                break;
terminate;
```

void cart(string filename)

```
declare line as string;
open filename as printfile;
if printfile is open:
        if the printfile.peek equal -1
                output not recorded found;
        else
                while getting each line from printfile using getline
                        output line;
        close the printfile;
else
        output file is not found;
terminate;
```

bool edit_cart()

```
declare cart_option as integer and b as boolean;
ask user to input cart_option and save as cart_option;
switch on cart_option
case1:
        declare delete_id, delete_seat as string;
        ask user to input delete_id and delete_seat and store them each;
        If cin is false
                Clear the remain input;
        call delete_id_seat(delete_id, delete_seat);
        return equal true;
        break;
case2:
        set b equal function exit_page() and call it;
        return b;
        break;
default
```

if not cart_option
clear the remaining input;
output invalid input ask users input again;
otherwise cart_option is not equal 1 and cart_option is not equal 2
output invalid input ask users input 1 or 2;
return true;
terminate;

void delete_id_seat(string delete_id, string delete_seat)

declare num_book_in_cart as integer and get number of booking from cart;

declare arrays_cart_array and ticket_book as string, size as 500 and store the content of cart;

declare delete_seat_code as string, valid_info as Boolean and equal false;

open cart.txt and read the content to infile;

start a for loop as I smaller than num_book_in_cart I increase1

declare line as string;

read the line from infile and store it into cart_array[i] using getline;

open booking.txt and read the content into infile2;

declare count_line as integer equal 0, and line1 as string.

While there have lines at infile2

Read the line from infile2;

count_line increase1;

clear infile2 and reset the file stream to beginning;

start a for loop as j smaller than count_line j increase 1

read the line from infile2 and store it into ticket_book[i];

if delelte_id and delete_seat are in correct formal and length

open cart.txt for outfile and booking.txt for outfile2 both in write mode;

start a for loop as k smaller than num_book_in_cart k increase 1

if the record_id and record_seat of current item don't contain the delte_id and delete_seat

call delete_id_seat_code() as delete_seat_code;

write the item into cart file;

start a for loop as l smaller than count_line, l increase 1;

if delete_seat_code is not found at ticket_book

write ticket_book into outfile2;

set valid_info as true;

close outfile and outfile2;

if valid_info is false

    output the thing you selected no found;

close infile and infile2;

terminate;

string delete_id_seat_code(string delete_seat, string delete_id)

declare delete_seatcode as character and size is 3;

start a for loop j to 3 j increase 1 every time

    set delete_seatcode[j] = delete_id[j];

set number_of_delete_id = convert first and second characters of delete_id to int and add them together;

set number = convert number_of_delete_id to char;

set delete_seat_code = concatenate number and delete_seat;

return delete_seat_code;

terminate;

Bool payment()
Call print4();
line_in_cartfile = array_cart(carts);
display time ;
discount_rate = discount();
member_discount_rate = member_discount();
display the contents of the cart.txt;
declare subtotal_price as double and equal to 0;
declare sen as a integer;
start a loop for i from 0 to line_in_cartfile and plus 1 every time
    subtotal_price += carts[i].record_price;
calculate the discount price with subtotal_price * discount_rate;
calculate the member discount price with subtotal_price * member_discount_rate;
calculate the total price with subtotal_price - round((discount_price * 100)) / 100 -
round((member_discount_price * 100)) / 100;
obtain the last two digits of the totals price with sen = round(fmod(total_price * 100, 10));
if sen % 10 bigger than or equal 5 then sen += 10;
sen equal sen - (sen % 10);
calculate total_price_after_round with total_price - (fmod(total_price * 100, 10) / 100) + (sen / 100.0) //
round the total price;
display the details of prices;
declare find_change as double and equal to 0;

find_change = payment_method(total_price_after_round)
declare transaction_id and filename as string;
declare num_of_file as integer equal 0;
start a loop for i from 0 to 500 and plus 1 every time
    if i is between 0 and 9, set transaction_id = “000” + i as a string;

if i is between 10 and 99, set transaction_id = “00” + i as a string;
if i is bigger than or equal 100, set transaction_id =”0” + i as a string;
set filename equal “transacrtion_id_” + transaction_id +”.txt”;
open the file with filename and store the data in receipt_in_file;
if the file was opened, number_of_file increase1;
if the file open is false
    open the file with filename and store the data in receipt_in_file;
    write the heading into file;
    open the cart.txt file in reading mode and declare line as string;
    while there are still have lines to read in cart
        read the line from cart using getline;
        write the line into receipt_in_file;
    write the following details into file;
    if method equal “E-wallet” or “Credit Card”, set customer_pay equal total_price;
    write the following details and thankyou into file;
    close the cart;
    close the receipt_in_file;
    close the count_file;
    open "history.txt" in append mode as history_file
    if history_file is open:
        write transaction_id, customer_pay, and method into history_file;
    close the history_file;
    break and exit the loop;
open the cart.txt file with delete_content_of_file in write mode
close the cart.txt file;
declare a as a Boolean and equal to true;
declare print_receipt as character;
as long as a is true
    ask user to input and store the data into print_receipt;
    clear the remaining input;
    if print_receipt is ‘y’ or ‘Y’
        open the file with filename and read the data in print
        declare the line as string;
        while there are still have lines to read in print
        read the line from print using getline and print it out the line;
        set a equal true
        close the print;
    else
        set a equal false;
        return true;
terminate

double discount()
declare line_in_cartfile as integer;
line_in_cartfile equal array_cart(carts);
output number of ticket(s) booking with line_in_cartfile
if line_in_cartfile bigger than or equal 4 and line_in_cartfile smaller than 7

```
        return 0.05;
otherwise line_in_cartfile bigger than or equal 7
        return 0.1;
else
        return 0.0;
Terminate;
```

double member_discount()

```
declare yes_no as a character, customer_id, password, membership as string;
declare a and valid_member as boolean equal true;
as long as a is true;
        ask user to input and store it into yes_no
        clear the remaining input
        if yes_no is Y or y
                ask user to input customer_id and password then store them each;
                clear the remaining input;
                open open User_Details.txt as member_file;
                declare line as string;
                While getting each line from member_file using getline
                        Create a new instance of the struct User name user;
                        user.id = read line from file;
                        user.password = read line from file;
                        user.name = read line from file;
                        user.nric = read line from file;
                        user.email = read line from file;
                        user.phone = read line from file;
                        user.membership_level = read line from file;
                        if (customer_id equal user.id and password equal user.password)
                                membership equal user.membership_level;
                                if membership equal “gold”
                                        return  0.05;
                                otherwise membership equal “platinum”
                                        return 0.1;
                                otherwise membership equal “diamond”
                                        return0.15;
                        else
                                valid_member equal false;
                close the member_file;
                if valid_member equal false
                        output not member;
                        a equal true;
        else
                return 0.0;
Terminate;
```

double payment_method(double total_price_after_round)

```
declare a and b as a Boolean equal true, reset_payment as character;
```

```
declare find_change as double equal 0;
as long as a is true;
        as user to input and store it as method;
        if method equal “E”
                set method equal “E-wallet”;
                set find_change equal 0;
                set b to true;
        otherwise if method equal “C”
                set method equal “cash”
                ask user to input and store it in customer_pay;
                If customer_pay is false
                        Clear all the remaining input;
                        Output invalid;
                        Set a equal true, and b equal false;
                Otherwise if customer_pay is true
                        find_change equal customer_pay – total_price_after_round;
                        Set b equal true;
        Otherwise if method equal “CC”
                Set method equal “Credit Card”;
                Find_change equal 0;
                Set b equal true;
        Else
                Output unacceptable method;
                Set a equal true, and b equal false;
        If find_change smaller than 0 and method equal as “Cash”
                Output please pay again;
                Set a equal true, and b equal false;

As long as b is true
        Ask user to input and store in reset_payment;
        Clear all the remaining input;
        If reset_payment is ‘y’ or ‘Y’
                Output find_change;
                Set b and a equal false;
        Else
                Set b equal false, and a equal true;
Return find_change;
 Terminate;
```

Function: void history_list()

```
Declare number as integer and equal 0;
Output history record heading;
Open “history.txt” as myfile;
If the myfile is open:
        Declare line, id, price, payment_method as string;
                While read a word from history.txt file and store it in id;
                Read a word form history.txt file and store it in price;
```

```
                Read the remain word of the line from history.txt file and store it in payment_method;
                Output id, price, payment_method;
        End while
        Close the myfile;
Else
        Output unable open the file;
Terminate.
```

Function: void history()

```
Declare found as a Boolean and set to true;
Declare filename and transaction_id_search as string;
Repeat
        Clear screen;
        Call history_list();
        Ask user to input transaction_id_search;
        If the input of transaction_id_search are digit and the length is 4
                Set filename = "transacrtion_id_" + transaction_id_search + ".txt";
                Open filename as myfile;
                If myfile is open:
                        While getting each line from myfile using getline
                                Output line;
                                Set found to true;
                        End while
                        Close myfile;
                Else
                        Display not found;
        Else
                Clear input buffer;
                Output invalid input;
        If not exit page():
                transaction_id_search = “Q”;
until transaction_id_search = “Q”
return true;
Terminate.
```

Function : bool option4()

```
Declare continue_program as bool and equal to true;
As long as the continue_program is true;
    Display membership interface;
    Display “Option:”;
    Declare menu_option as int;
    Read input from user and store it in menu_option;
    Clear input buffer;
    If menu_option is 1;
      call acc_registration function;
      clear the screen;
   Else if menu_option is 2;
```

```
        call acc_management function;
        clear the screen;
    Else if menu_option is 3;
        set continue_program to exit_page function;
    Else
        Display "Invalid option! Please enter a number between 1 and 3.";
    Return true;
```

Function : bool id_used(string user_id)

```
    Open file("User_Details.txt");
    Declare line as string;
    While (there are lines left in the file) do
        If line is same as user_id;
            Close the text file;
            Return true;
    Close the text file;
    Return false;
```

Function : bool nric_used(string user_nric)

```
    Open file("User_Details.txt");
    Declare line as string;
    While (there are lines left in the file) do
        If line is same as user_nric;
            Close the text file;
            Return true;
    Close the text file;
    Return false;
```

Function : bool email_used(string user_email)

```
    Open a file named "User_Details.txt";
    Declare line as string;
    while (getline(file, line))
        If line is same as user_email;
            Close the text file;
            Return true;
    Close the text file;
    Return false;
```

Function : bool phone_used(string user_phone)

```
    Open a file named "User_Details.txt";
    Declare line as string;
    While (there are lines left in the file) do
        If line is same as user_phone;
            Close the text file;
            Return true;
    Close the text file;
    Return false;
```

Function : void acc_registration()

```
Declare user_id, password, confirm_password, full_name, nric, email, phone_number and
membership_level as string, is_valid and valid_level as bool, valid_level is false;
Display Registration Menu;
As long as the condition is true;
      Display confirmation question;
      Declare option as string;
      Read input from user and store it in option;
      If  'option' is 'n' or 'N'
            Return;
      Else If 'option' is 'y' or 'Y'
            Exit loop;
      Else
            Display "Invalid input! Please try again.";
Do
     Display "ID:";
     Read input from user and store it in 'user_id';
     If 'user_id' is already in use;
          Display information about invalid input;
          Set 'is_valid' is equal to false;
     Else
          is_valid is equal to validate_user_id(user_id);
 While 'is_valid' is false;

Do
     Display "Password:";
     Read input from user and store it in 'password';
     Set is_valid is equal to validate_password(password);
While 'is_valid' is false;

Do
     Display " Confirm Password:";
     Read input from user and store it in 'confirm_password';
     Set is_valid is equal to validate_confirm_password(password, confirm_password);
While 'is_valid' is false;

Do
     Display " Full Name:";
     Read input from user and store it in full_name';
     Set is_valid is equal to validate_full_name(full_name);
While 'is_valid' is false;

Do
     Display " NRIC:";
     Read input from user and store it in 'nric';
     If 'nric' is already in use;
```

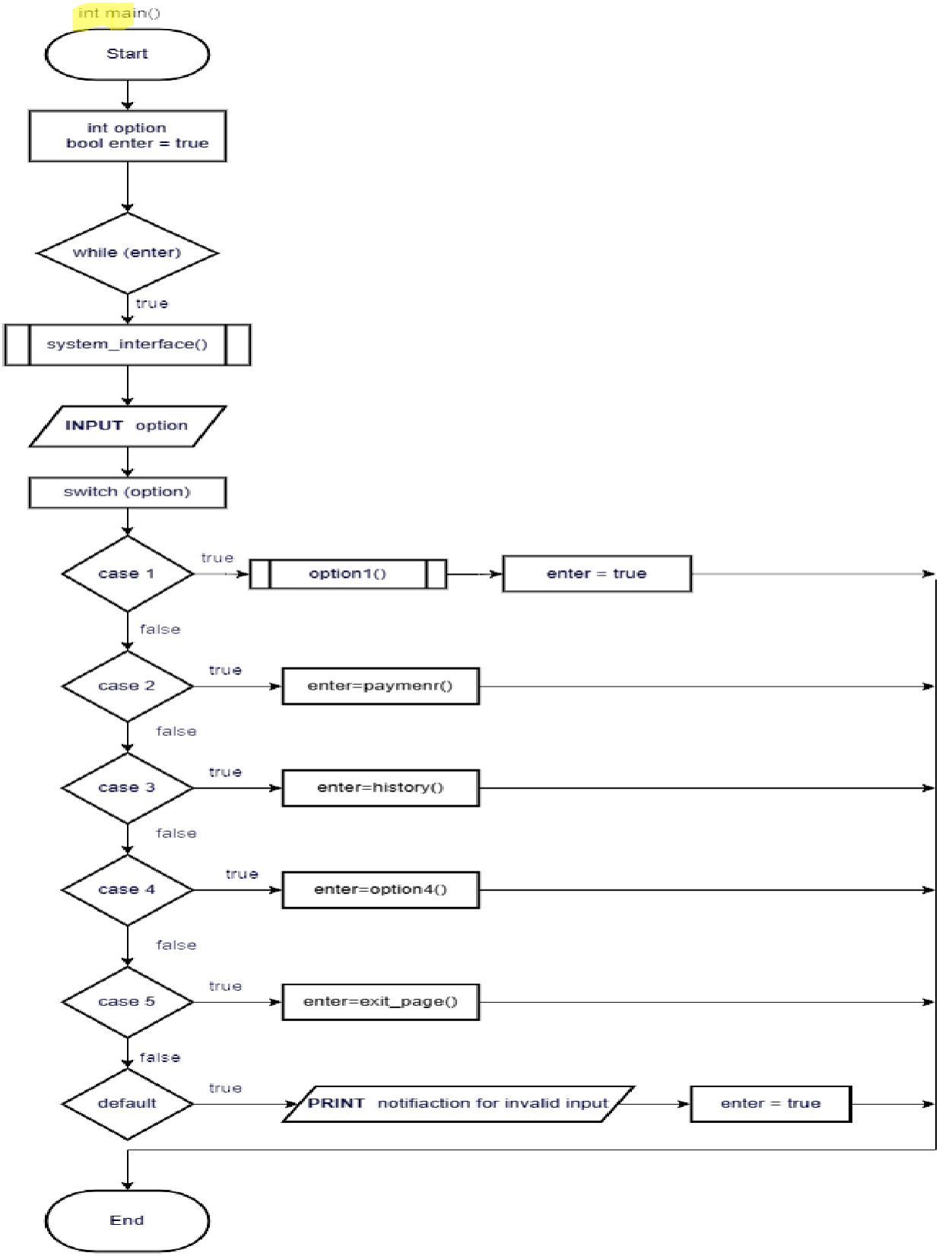
```
            Display information about invalid input;
            Set 'is_valid' is equal to false;
        Else
            is_valid is equal to validate_user_id(user_id);
    While 'is_valid' is false;
Do
        Display "Email:";
        Read input from user and store it in ' email';
        If 'email' is already in use;
            Display information about invalid input;
            Set 'is_valid' is equal to false;
        Else
            is_valid is equal to validate_email(email);
    While 'is_valid' is false;
Do
        Display " Phone Number:";
        Read input from user and store it in 'phone_number';
        If ' phone_number' is already in use;
            Display information about invalid input;
            Set 'is_valid' is equal to false;
        Else
            is_valid is equal to validate_phone_number(phone_number);
    While 'is_valid' is false;

Do
        Display "Membership Level (Gold/Platinum/Diamond):";
        Read input from user and store it in 'membership_level';
        Set is_valid is equal to validate_membership_level(membership_level);
While 'is_valid' is false;

Open a file named "User_Details.txt" in append mode and store it in 'file';
If the file is open
        Write in all the user details;
        Close the file;
Else
        Display "Unable to open file!";
Display information about registered successfully;
Pause the program;
Return;
```

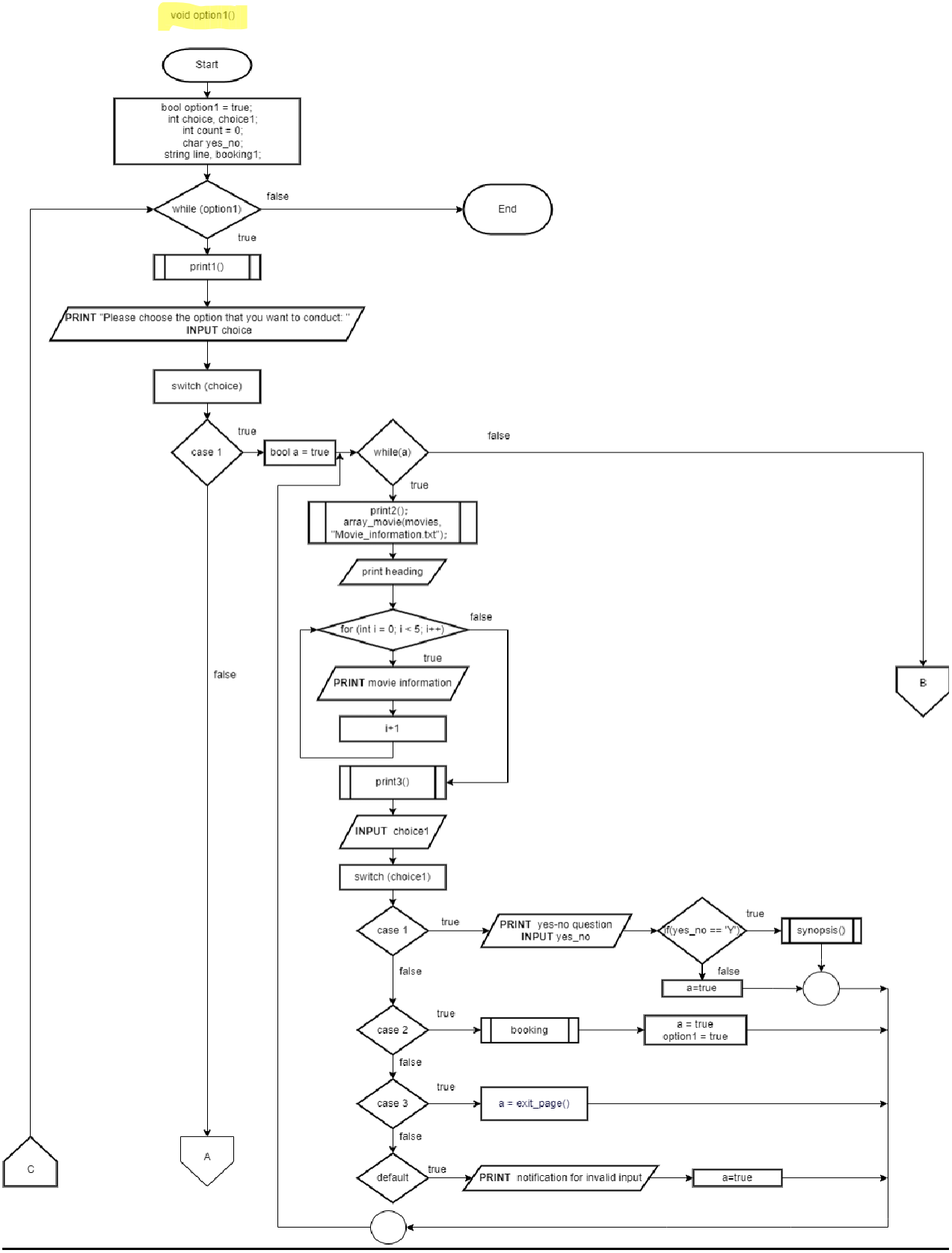
Function : void acc_management()

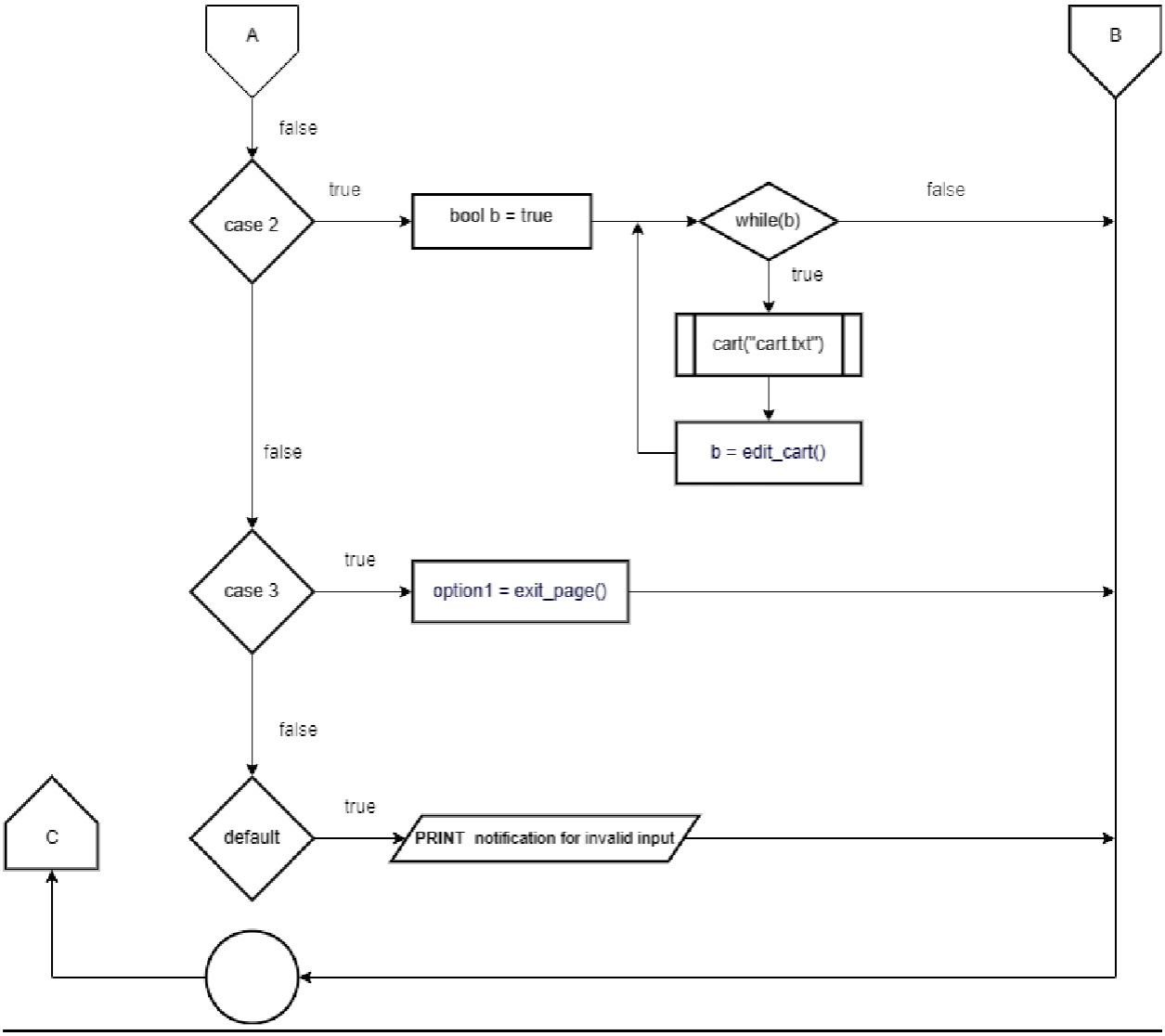
```
As long as the condition is true;
        Display confirmation question;
        Declare option as string;
        Read input from user and store it in option;
        If  'option' is 'n' or 'N'
            Return;
```

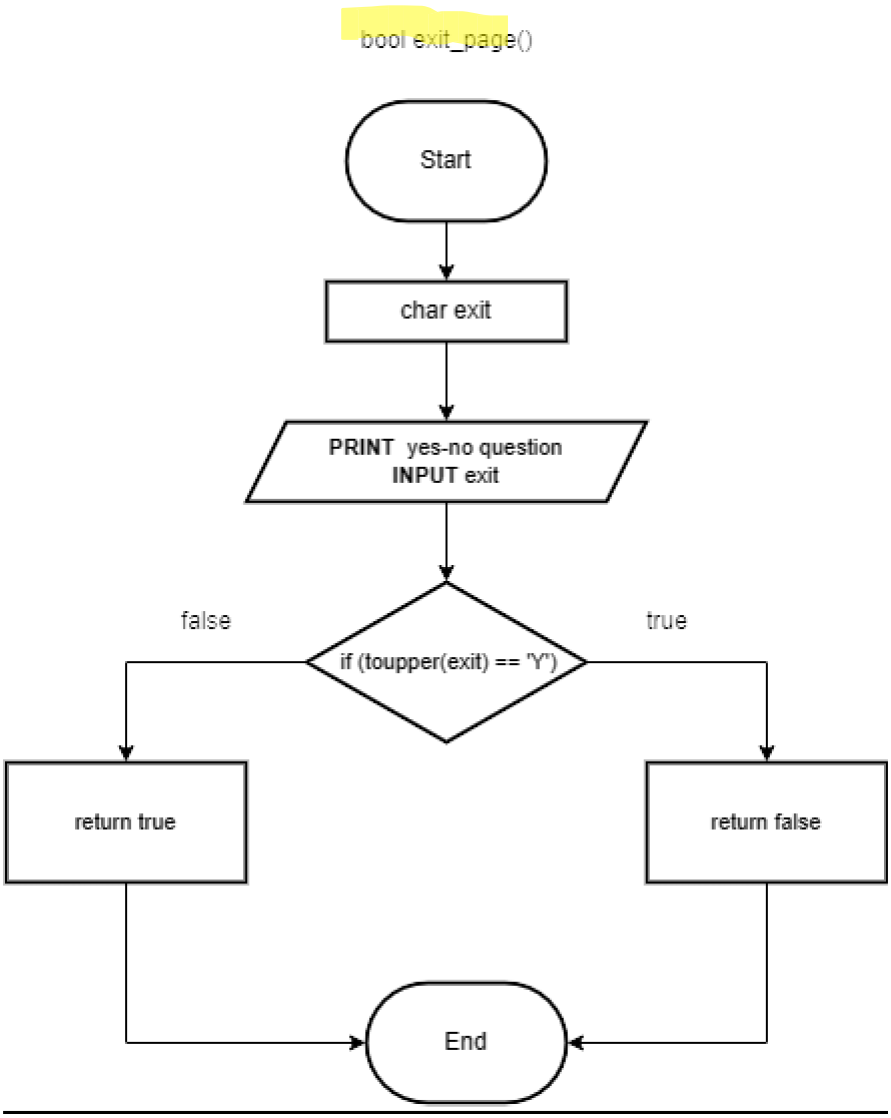
```
    Else If 'option' is 'y' or 'Y'
        Exit loop;
    Else
        Display "Invalid input! Please try again.";
Delcare loggedIn as Boolean and equal to false;

Open a file named "User_Details.txt" for reading purpose and store it in 'file';
If the file is open
    Display "Please log in to update your details.";
    Read ID from user and store it in 'id'
    Read password from user and store it in 'password'
    Declare ‘line’ as string;
    Open a file named "User_Details.txt" for writing purpose;
    While (there are lines left in the file) do
        Declare User user and user.id equal to line;
        Store the details from the file into the object;
        If the entered ID and password is same as those in the file
            Display user’s name and membership details;
            Set “loggedIn” to true;
            Declare new_value as string;
            Prompt the user to update their details;
            For each field do;
            Prompt the user for a new value for the field;
            If the user entered a value;
            Validate the value and update the corresponding field in the user object if it is valid;
            Display "Your details have been updated."
            Pause the program;
        Else
            Write the details of other users to the temporary file;
    Close the files;
    Remove a file named "User_Details.txt";
    Rename a file named "temp.txt" to "User_Details.txt";
If loggedIn is false;
    Display “Access denied.”;
    Pause the program;
    Return;
```

Flowchart

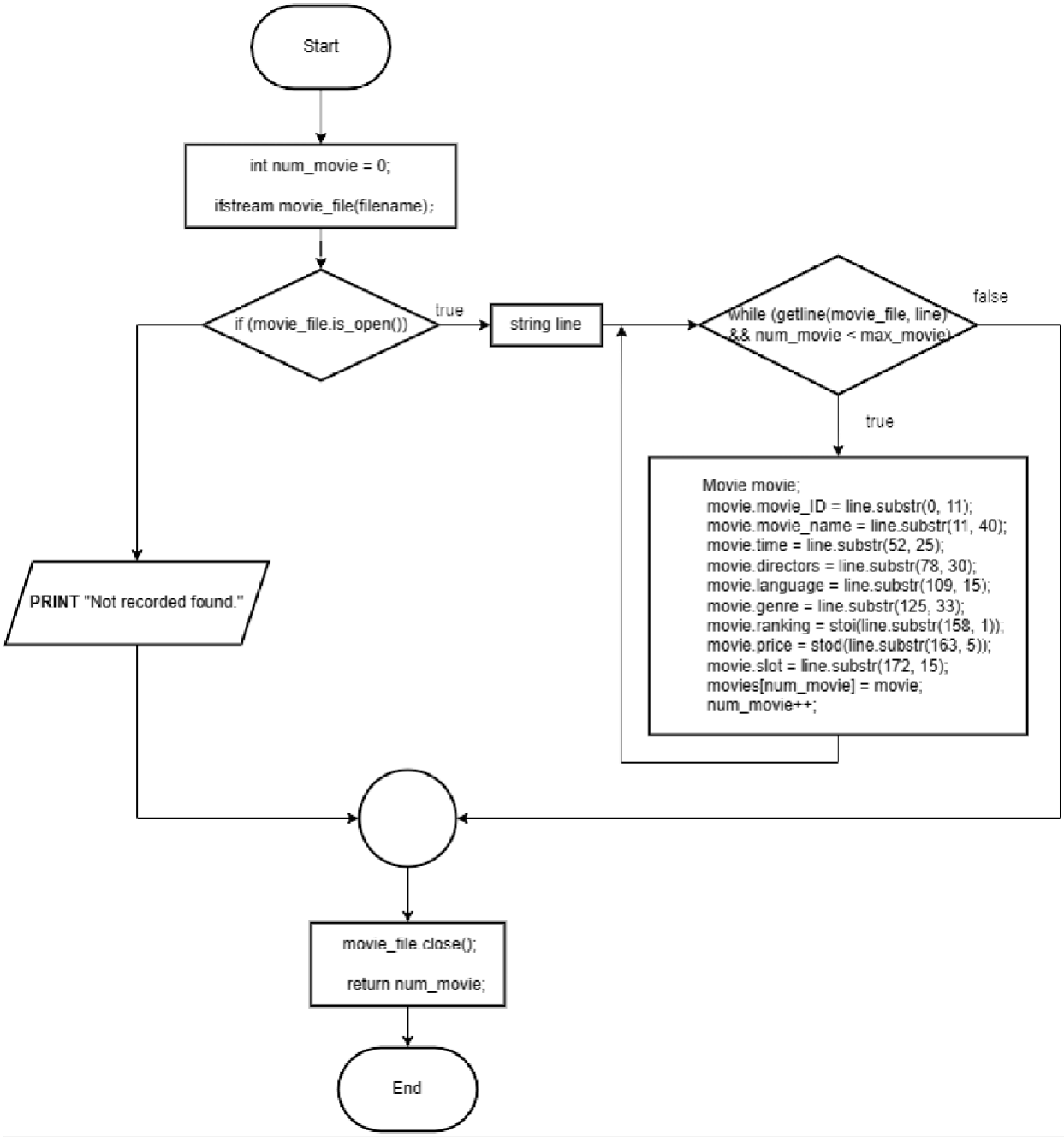

int main()
Start
int option
bool enter = true
while (enter)
true
system_interface()
INPUT option
switch (option)
case 1
true
option1()
enter = true
false
case 2
true
enter=paymenr()
false
case 3
true
enter=history()
false
case 4
true
enter=option4()
false
case 5
true
enter=exit_page()
false
default
true
PRINT notifiaction for invalid input
enter = true
End

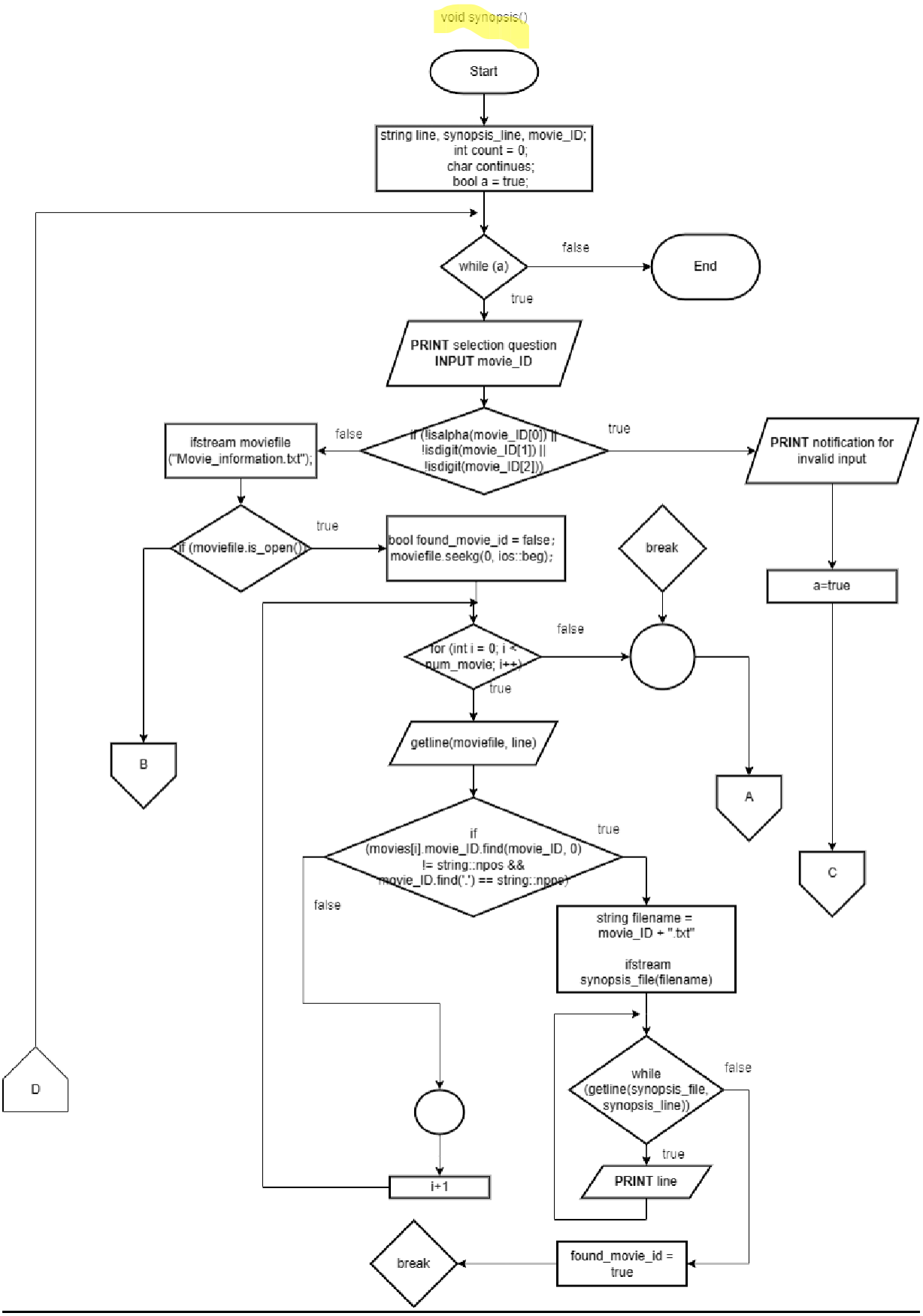

void option1()
Start
bool option1 = true;
int choice, choice1;
int count = 0;
char yes_no;
string line, booking1;
while (option1)
false
End
true
print1()
PRINT "Please choose the option that you want to conduct: "
INPUT choice
switch (choice)
case 1
true
bool a = true
while(a)
false
true
print2();
array_movie(movies,
"Movie_information.txt");
print heading
for (int i = 0; i < 5; i++)
false
true
PRINT movie information
i+1
B
false
print3()
INPUT choice1
switch (choice1)
case 1
true
PRINT yes-no question
INPUT yes_no
if(yes_no == 'Y')
true
synopsis()
false
a=true
case 2
true
booking
a = true
option1 = true
false
case 3
true
a = exit_page()
false
default
true
PRINT notification for invalid input
a=true
C
A

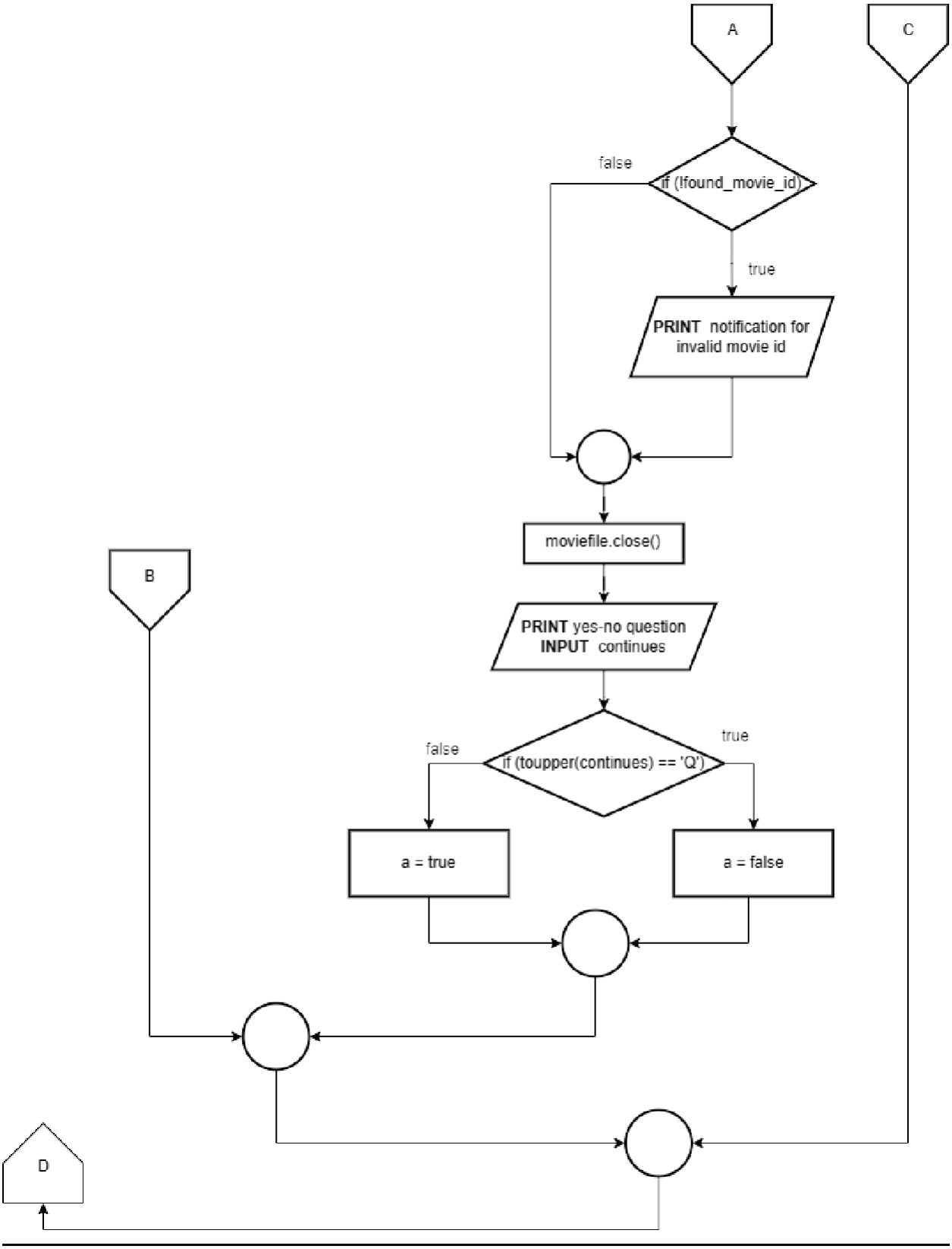

A
B
false
case 2
true
bool b = true
while(b)
false
true
cart("cart.txt")
b = edit_cart()
false
case 3
true
option1 = exit_page()
false
default
true
PRINT notification for invalid input
C

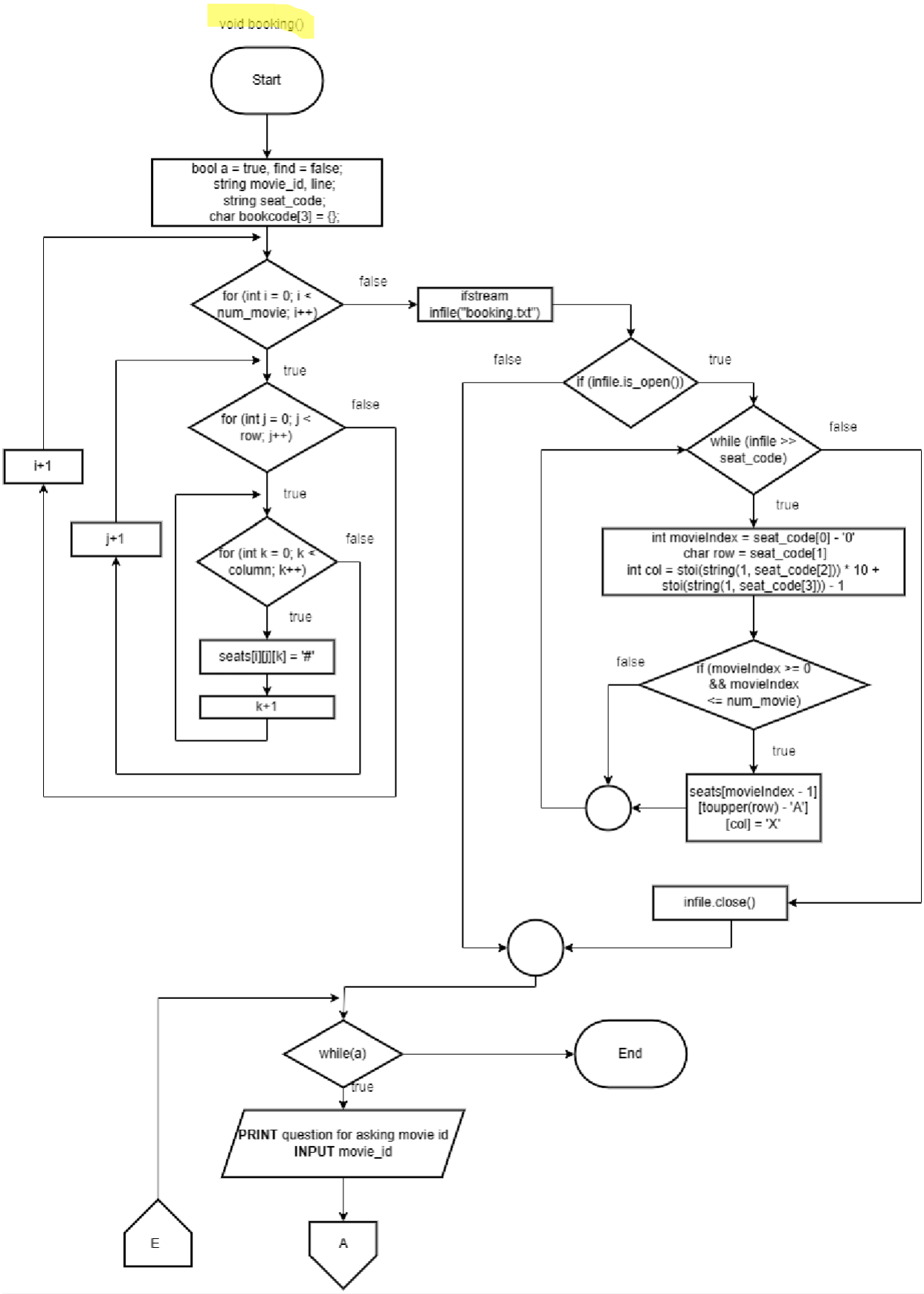

bool exit_page()
Start
char exit
PRINT yes-no question
INPUT exit
false
if (toupper(exit) == 'Y')
true
return true
return false
End

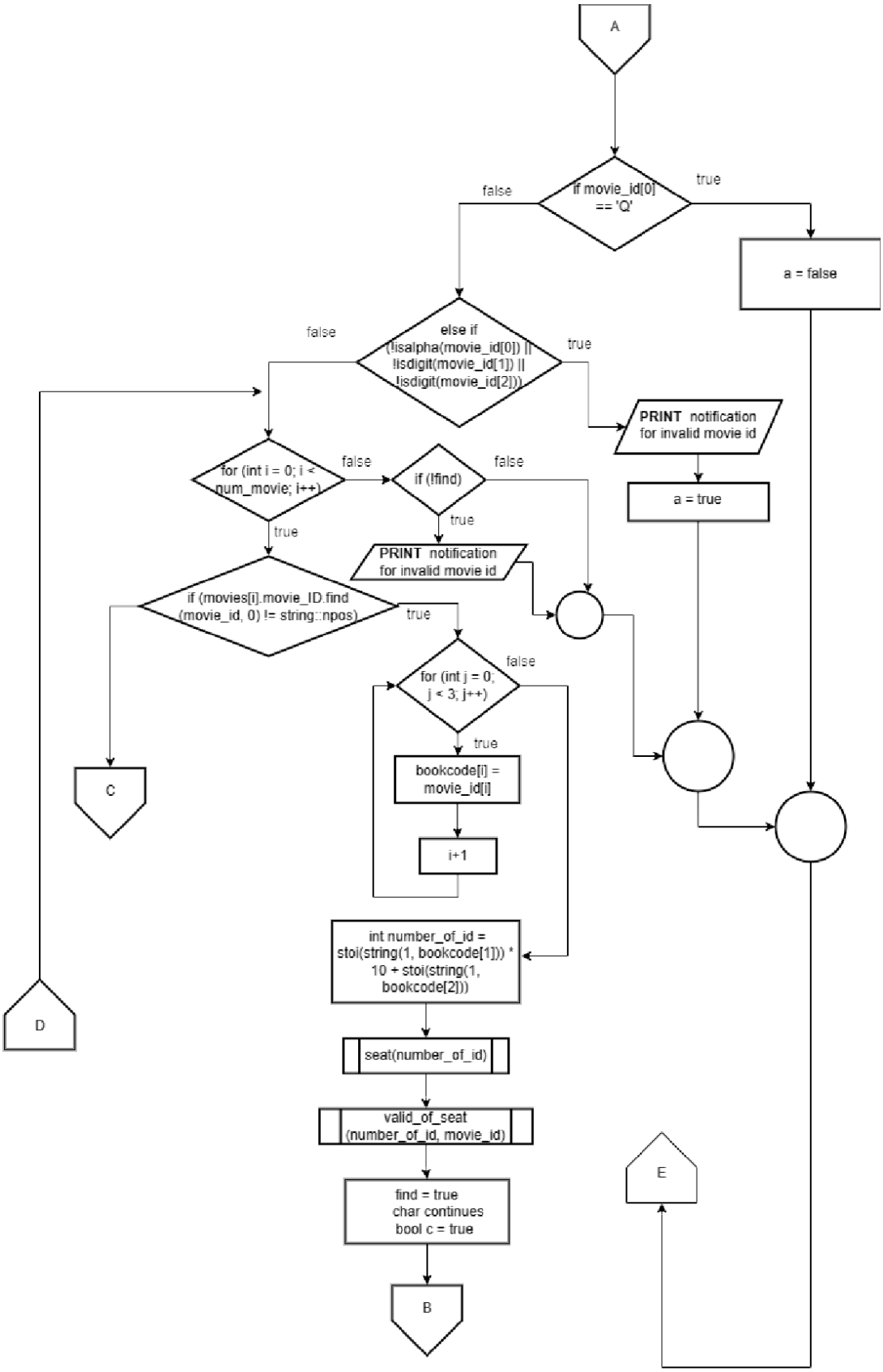
int array_movie(Movie movies[], const string& filename)

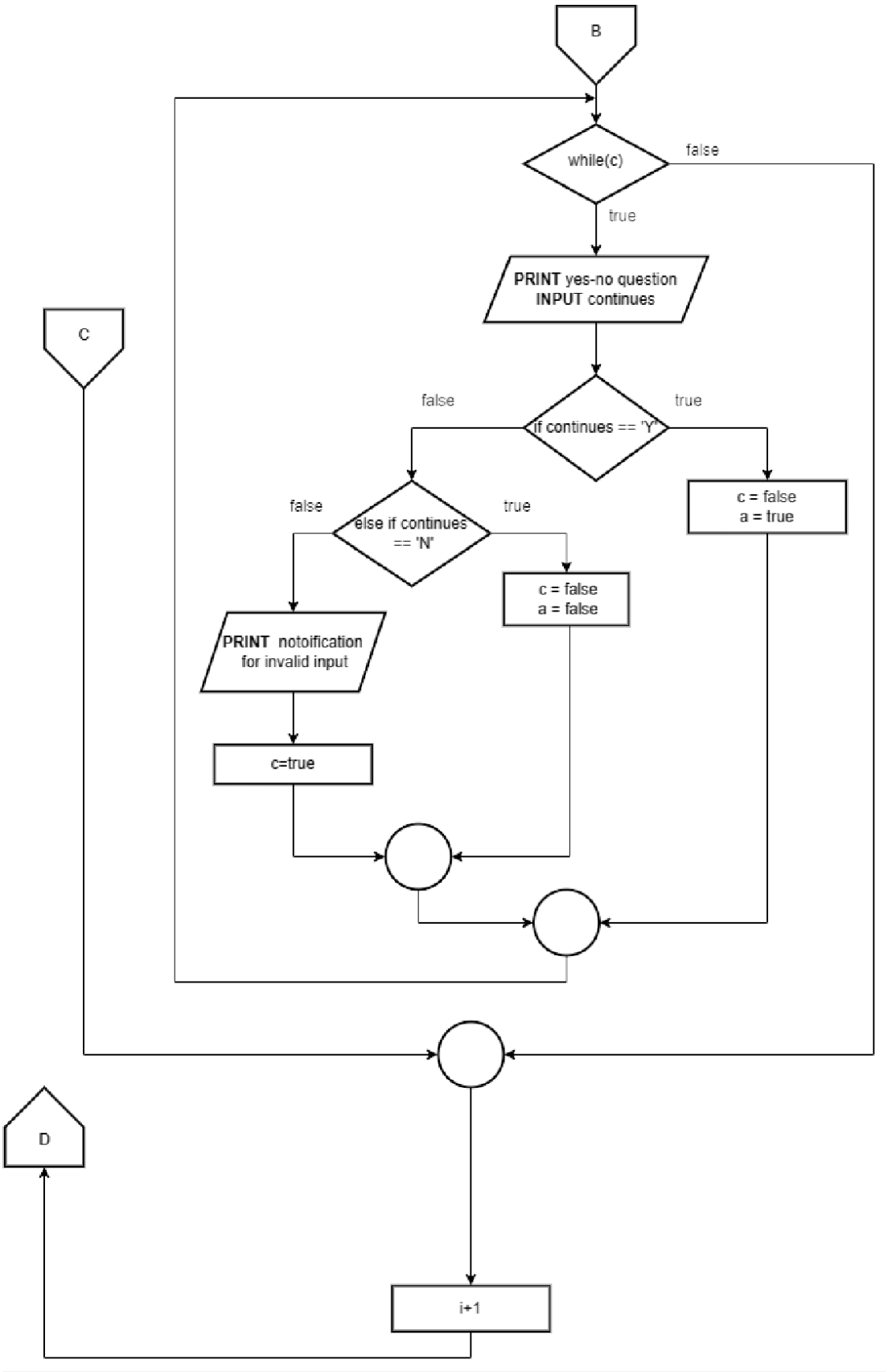



void synopsis()
Start
string line, synopsis_line, movie_ID;
int count = 0;
char continues;
bool a = true;
while (a)
false
End
true
PRINT selection question
INPUT movie_ID
if (!isalpha(movie_ID[0]) ||
!isdigit(movie_ID[1]) ||
!isdigit(movie_ID[2]))
false
true
ifstream moviefile
("Movie_information.txt");
PRINT notification for
invalid input
a=true
if (moviefile.is_open())
true
bool found_movie_id = false;
moviefile.seekg(0, ios::beg);
break
for (int i = 0; i <
num_movie; i++)
false
true
getline(moviefile, line)
B
A
C
D
if
(movies[i].movie_ID.find(movie_ID, 0)
!= string::npos &&
movie_ID.find('.') == string::npos)
true
false
string filename =
movie_ID + ".txt"
ifstream
synopsis_file(filename)
while
(getline(synopsis_file,
synopsis_line))
false
true
PRINT line
i+1
break
found_movie_id =
true

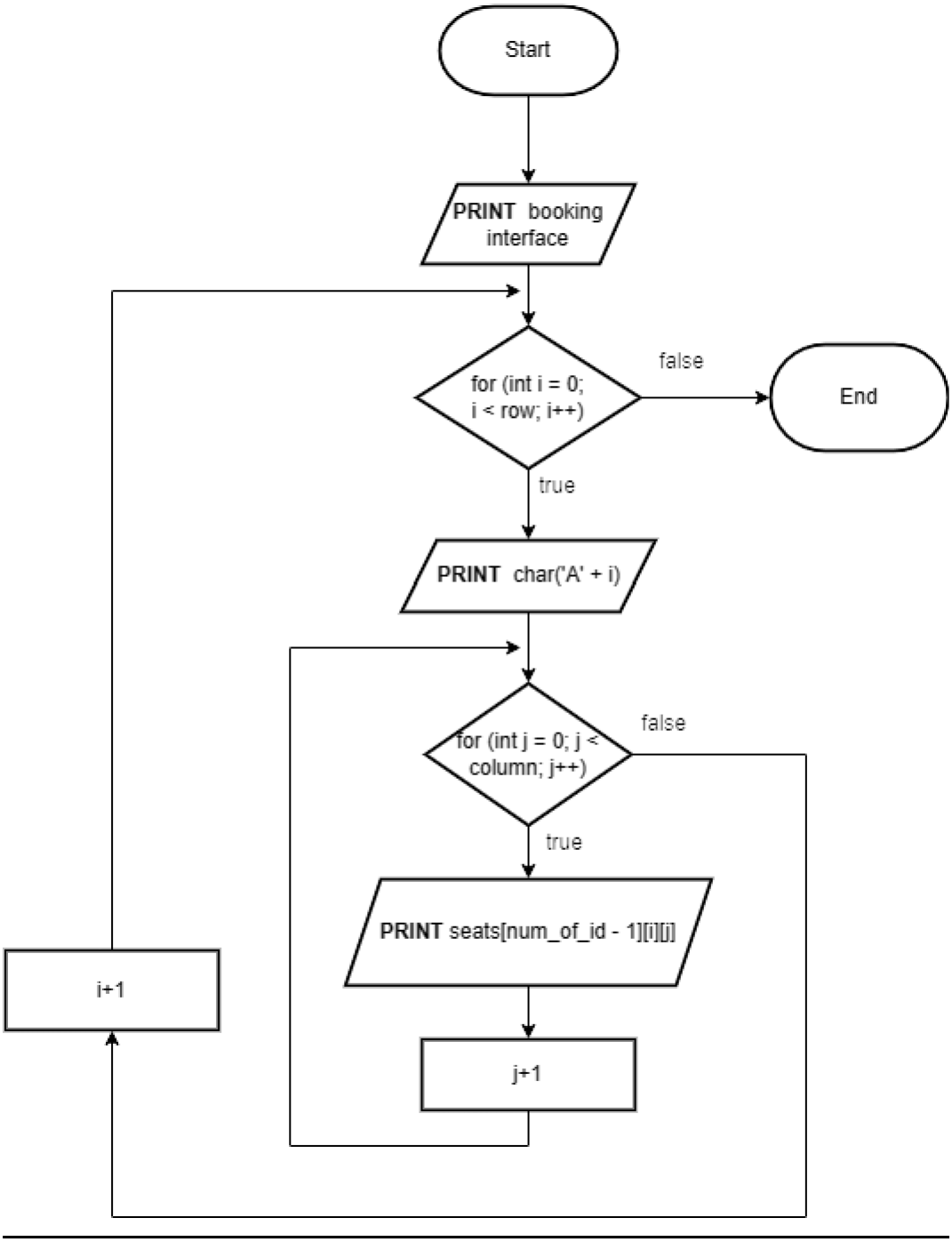

A
C
false
if (!found_movie_id)
true
PRINT notification for invalid movie id
moviefile.close()
B
PRINT yes-no question
INPUT continues
false
if (toupper(continues) == 'Q')
true
a = true
a = false
D

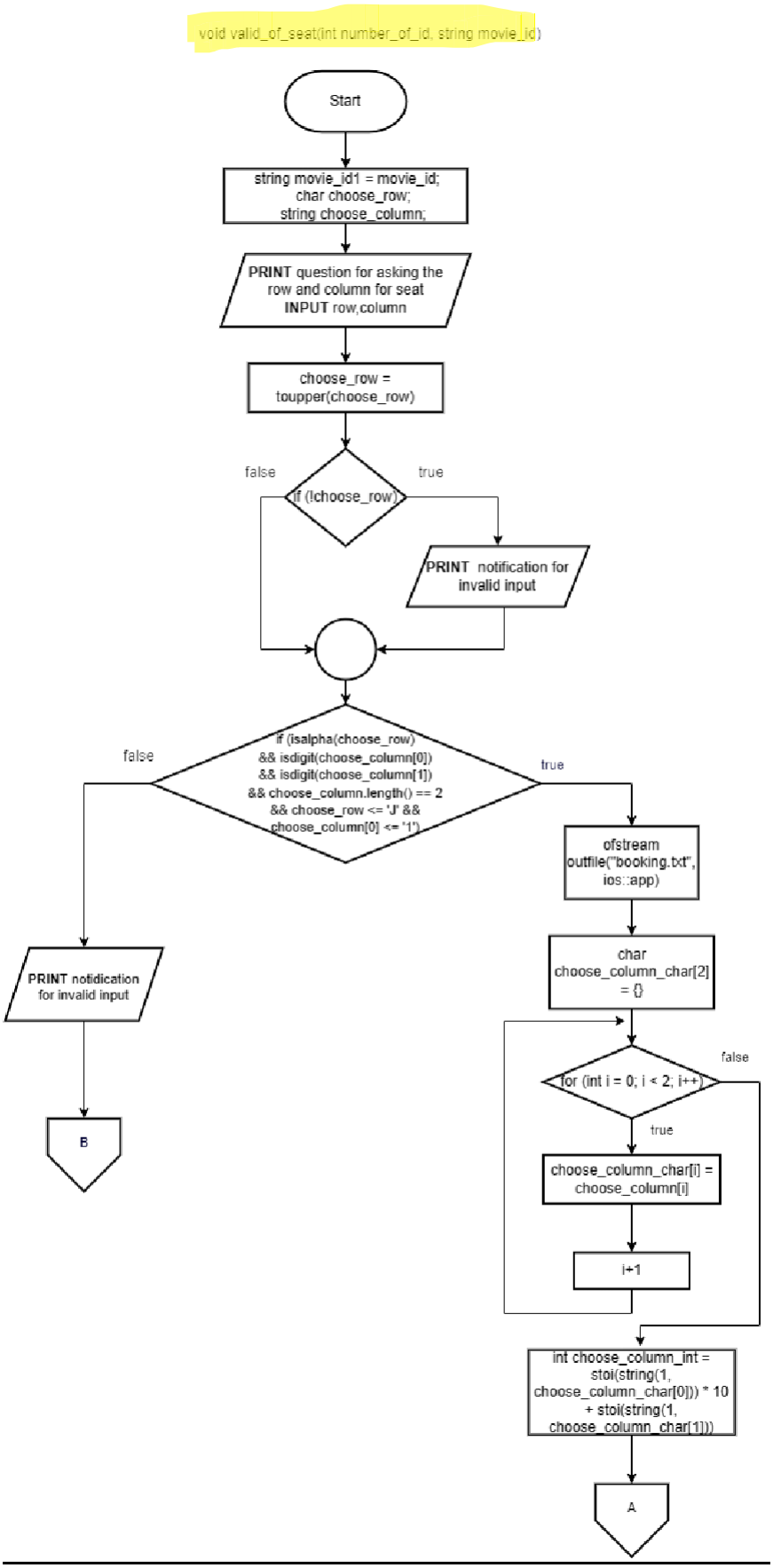

void booking()
Start
bool a = true, find = false;
string movie_id, line;
string seat_code;
char bookcode[3] = {};
for (int i = 0; i < num_movie; i++)
false
ifstream infile("booking.txt")
true
for (int j = 0; j < row; j++)
false
i+1
true
for (int k = 0; k < column; k++)
false
j+1
true
seats[i][j][k] = '#'
k+1
if (infile.is_open())
false
true
while (infile >> seat_code)
false
true
int movieIndex = seat_code[0] - '0'
char row = seat_code[1]
int col = stoi(string(1, seat_code[2])) * 10 + stoi(string(1, seat_code[3])) - 1
if (movieIndex >= 0 && movieIndex <= num_movie)
false
true
seats[movieIndex - 1] [toupper(row) - 'A'] [col] = 'X'
infile.close()
while(a)
End
true
PRINT question for asking movie id
INPUT movie_id
E
A

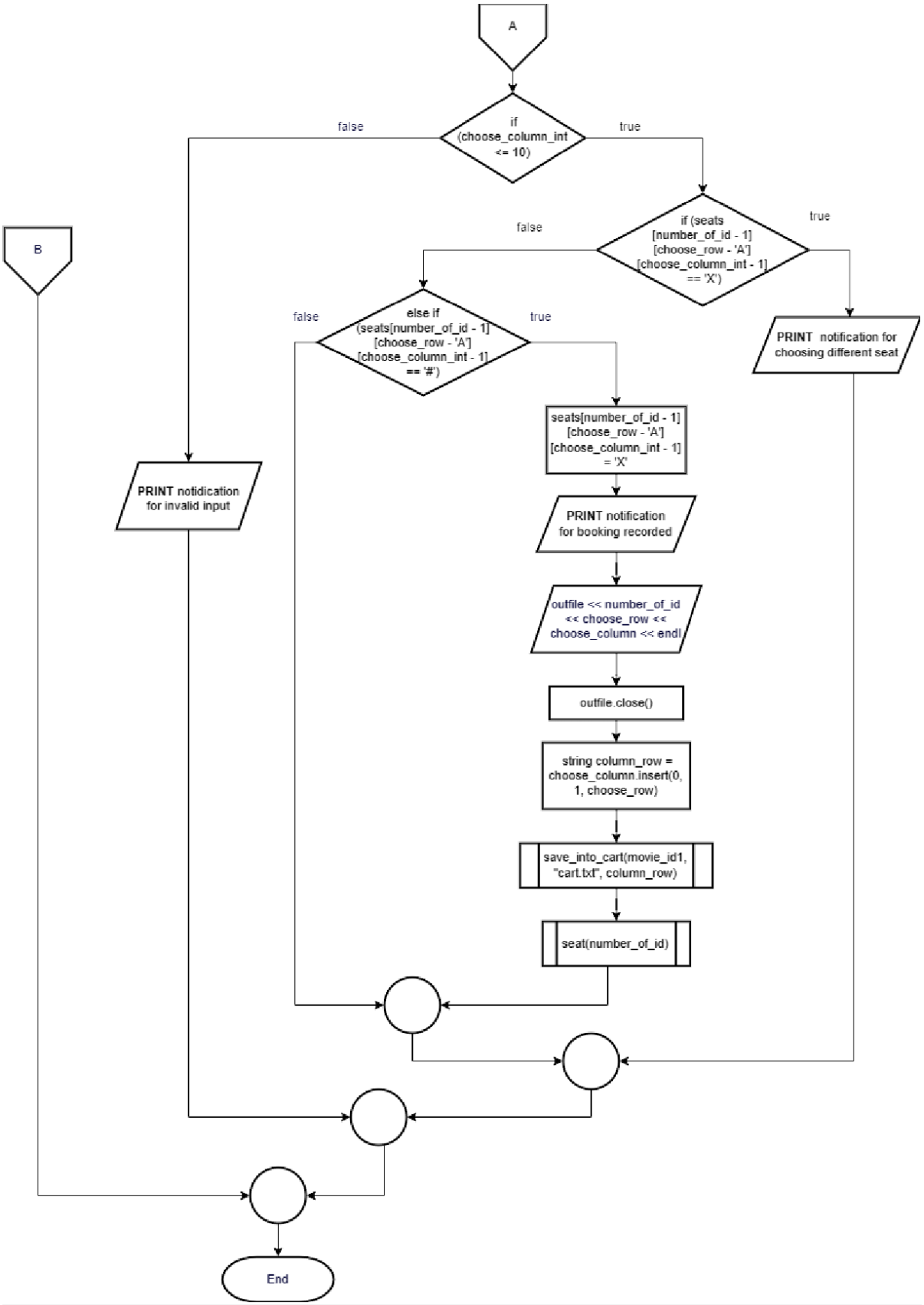

A
if movie_id[0] == 'Q'
false
true
a = false
else if (!isalpha(movie_id[0]) || !isdigit(movie_id[1]) || !isdigit(movie_id[2]))
false
true
PRINT notification for invalid movie id
a = true
for (int i = 0; i < num_movie; i++)
false
if (!find)
false
true
PRINT notification for invalid movie id
true
if (movies[i].movie_ID.find (movie_id, 0) != string::npos)
true
false
for (int j = 0; j < 3; j++)
true
bookcode[i] = movie_id[i]
i+1
C
D
int number_of_id = stoi(string(1, bookcode[1])) * 10 + stoi(string(1, bookcode[2]))
seat(number_of_id)
valid_of_seat (number_of_id, movie_id)
find = true
char continues
bool c = true
B
E

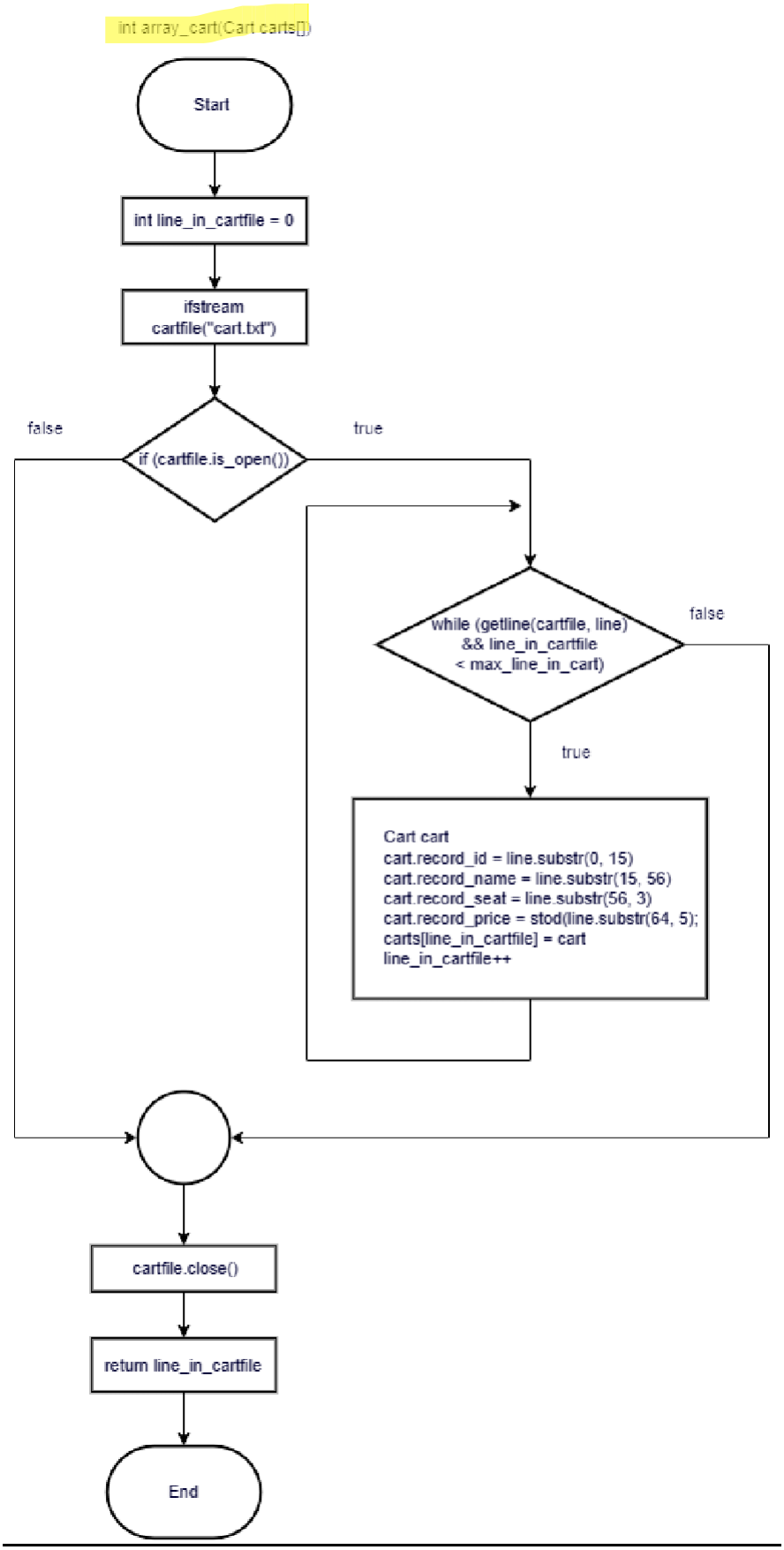

B
while(c)
false
true
PRINT yes-no question
INPUT continues
C
false
if continues == 'Y'
true
c = false
a = true
false
else if continues
== 'N'
true
c = false
a = false
PRINT notoification
for invalid input
c=true
D
i+1

void seat(int num_of_id)

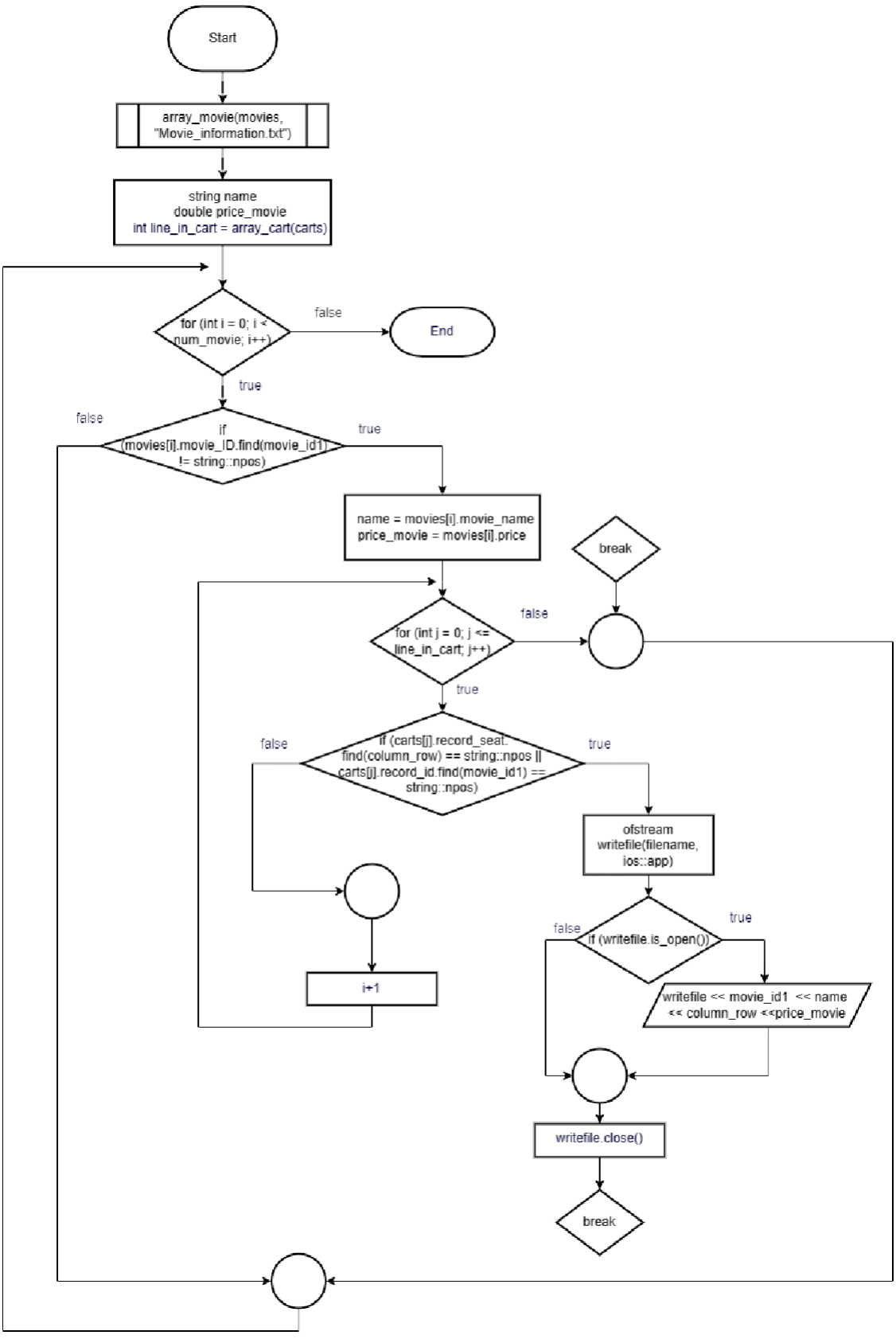


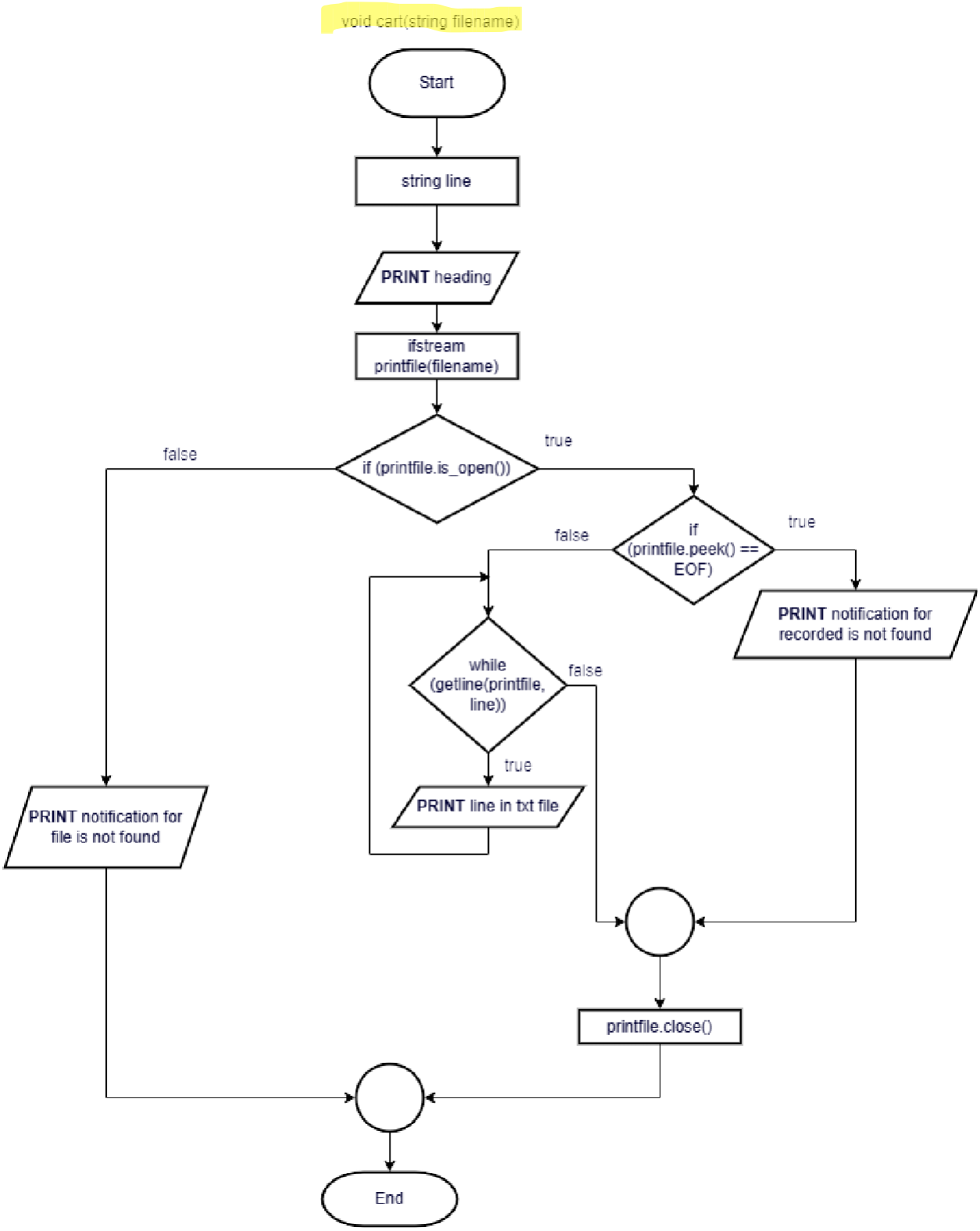

void valid_of_seat(int number_of_id, string movie_id)
Start
string movie_id1 = movie_id;
char choose_row;
string choose_column;
PRINT question for asking the row and column for seat
INPUT row,column
choose_row = toupper(choose_row)
false
true
if (!choose_row)
PRINT notification for invalid input
if (isalpha(choose_row) && isdigit(choose_column[0]) && isdigit(choose_column[1]) && choose_column.length() == 2 && choose_row <= 'J' && choose_column[0] <= '1')
false
true
PRINT notidication for invalid input
B
ofstream outfile("booking.txt", ios::app)
char choose_column_char[2] = {}
for (int i = 0; i < 2; i++)
false
true
choose_column_char[i] = choose_column[i]
i+1
int choose_column_int = stoi(string(1, choose_column_char[0])) * 10 + stoi(string(1, choose_column_char[1]))
A


A
if (choose_column_int <= 10)
false
true
if (seats [number_of_id - 1] [choose_row - 'A'] [choose_column_int - 1] == 'X')
true
false
B
else if (seats[number_of_id - 1] [choose_row - 'A'] [choose_column_int - 1] == '#')
false
true
PRINT notification for choosing different seat
PRINT notidication for invalid input
seats[number_of_id - 1] [choose_row - 'A'] [choose_column_int - 1] = 'X'
PRINT notification for booking recorded
outfile << number_of_id << choose_row << choose_column << endl
outfile.close()
string column_row = choose_column.insert(0, 1, choose_row)
save_into_cart(movie_id1, "cart.txt", column_row)
seat(number_of_id)
End

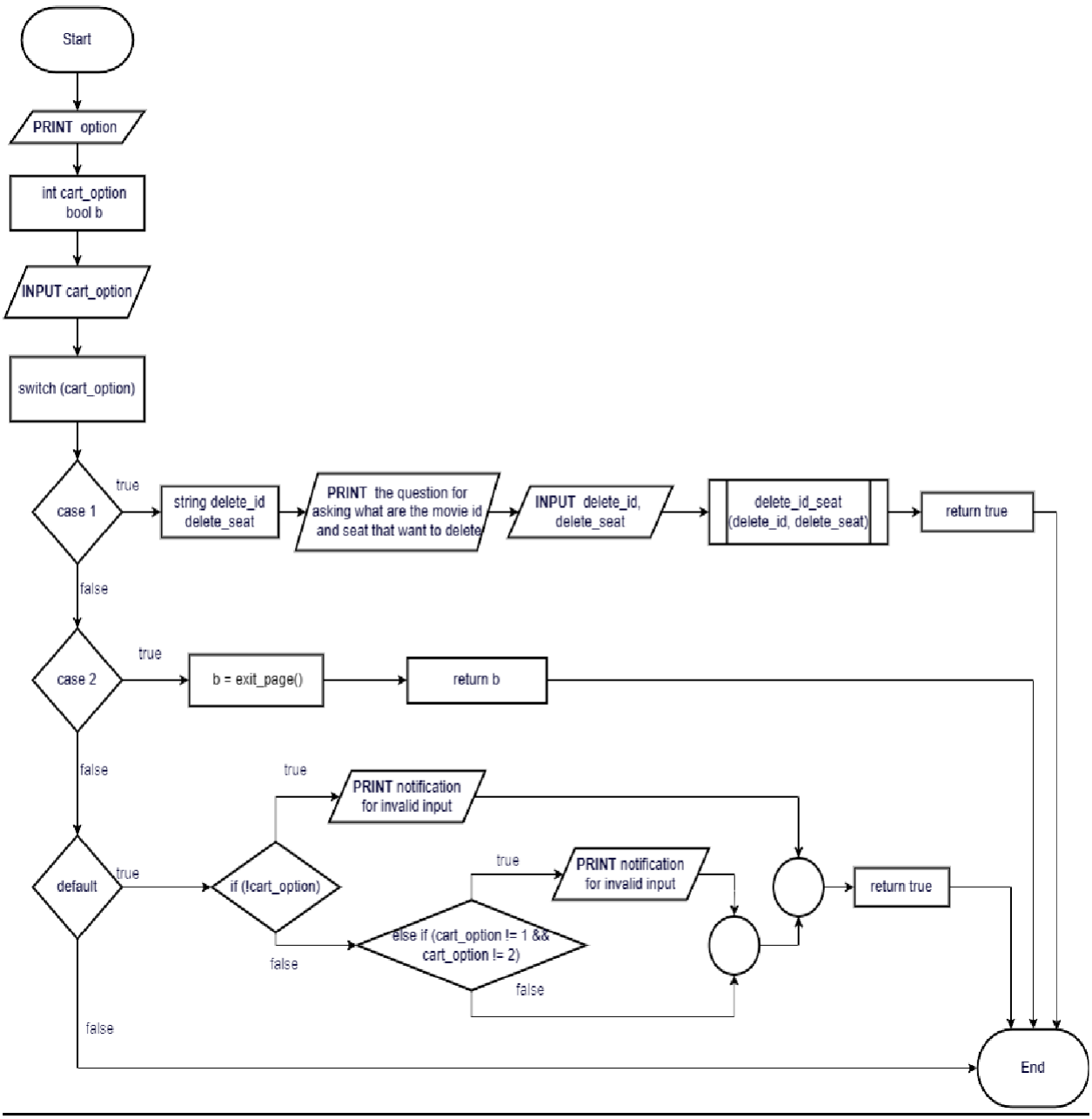

int array_cart(Cart carts[])
Start
int line_in_cartfile = 0
ifstream cartfile("cart.txt")
false
if (cartfile.is_open())
true
while (getline(cartfile, line) && line_in_cartfile < max_line_in_cart)
false
true
Cart cart
cart.record_id = line.substr(0, 15)
cart.record_name = line.substr(15, 56)
cart.record_seat = line.substr(56, 3)
cart.record_price = stod(line.substr(64, 5);
carts[line_in_cartfile] = cart
line_in_cartfile++
cartfile.close()
return line_in_cartfile
End

void save_into_cart(string movie_id1, string filename, string column_row)

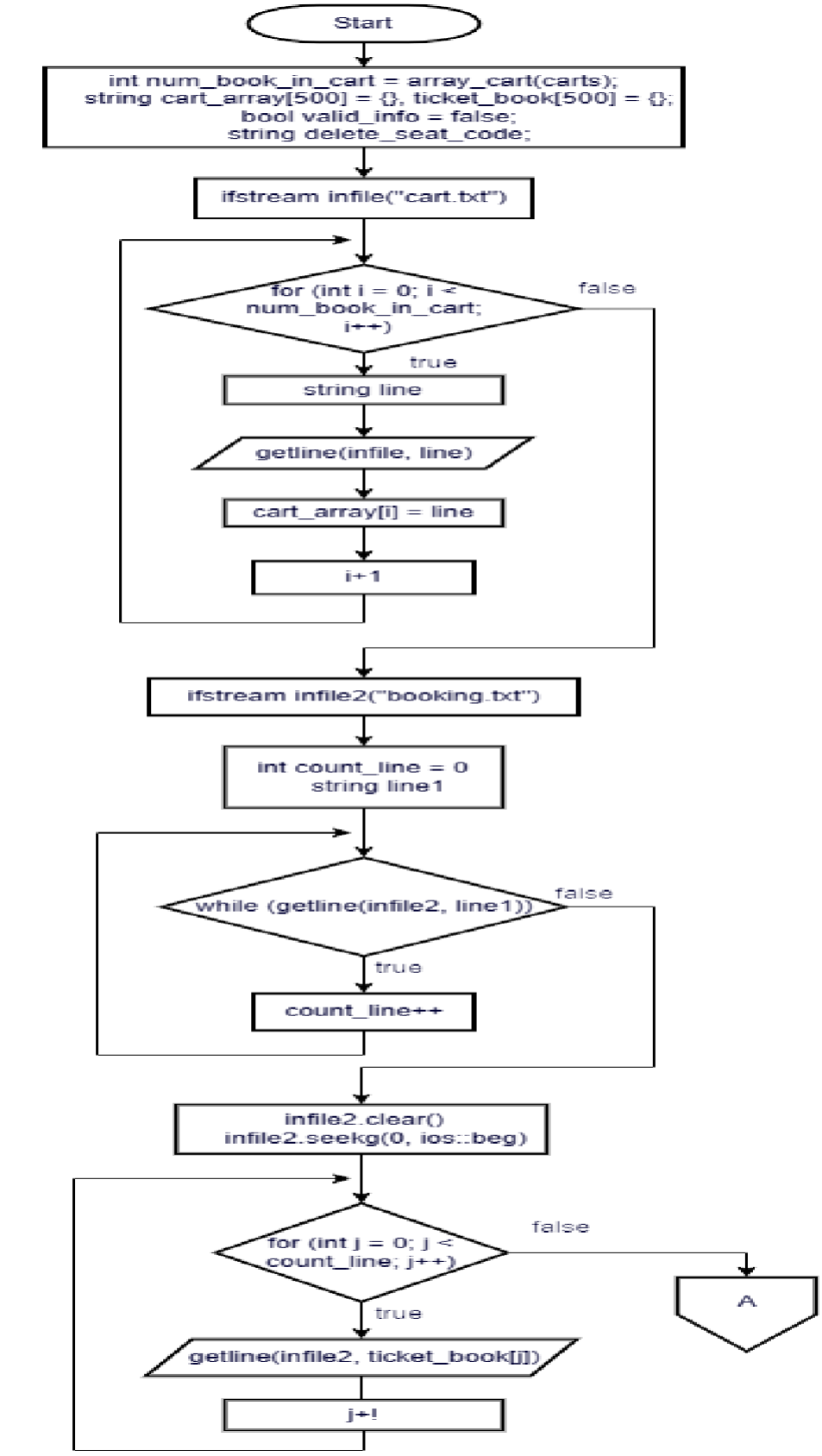


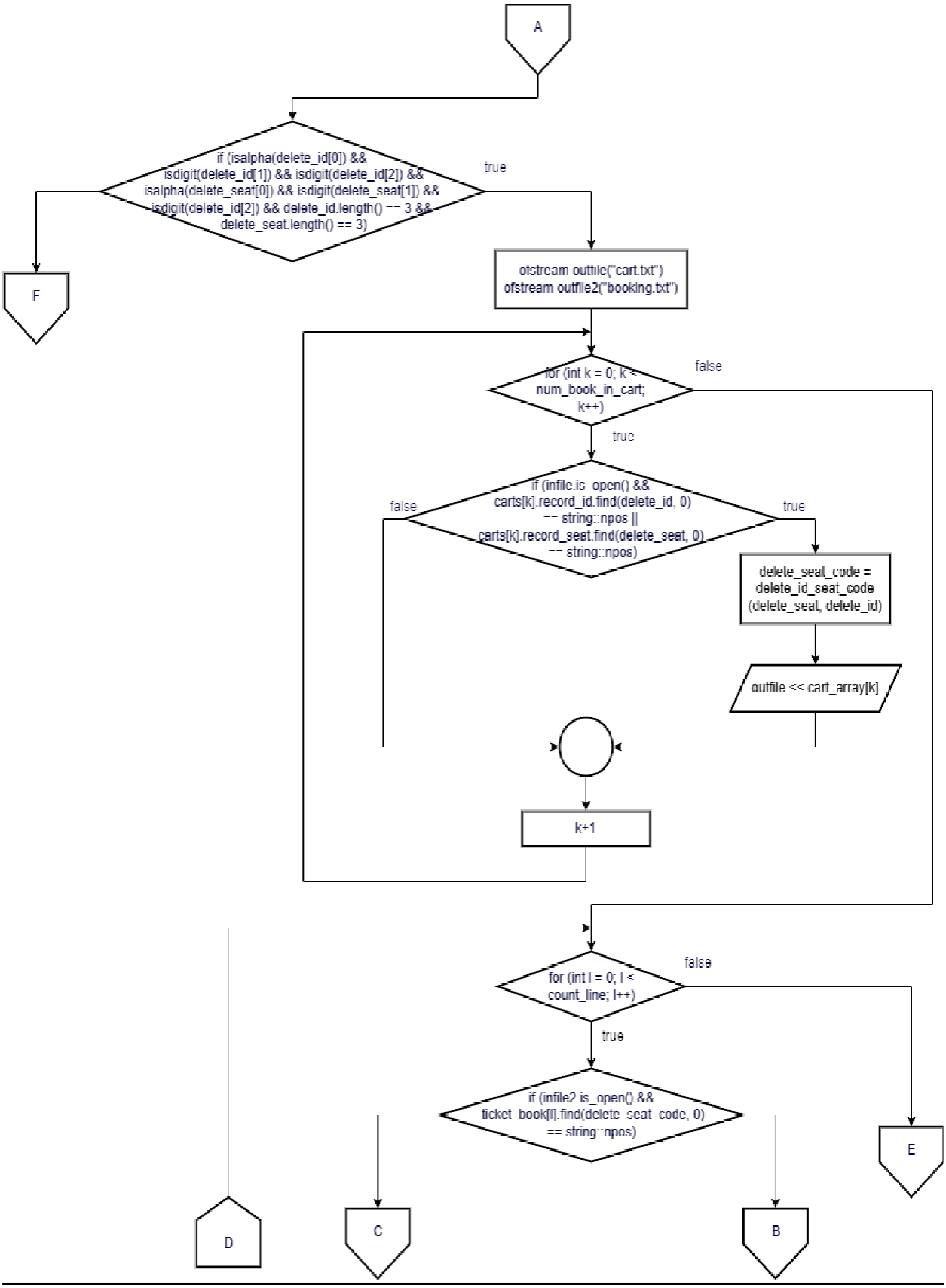

void cart(string filename)
Start
string line
PRINT heading
ifstream printfile(filename)
if (printfile.is_open())
false
true
if (printfile.peek() == EOF)
false
true
PRINT notification for recorded is not found
while (getline(printfile, line))
false
true
PRINT line in txt file
PRINT notification for file is not found
printfile.close()
End

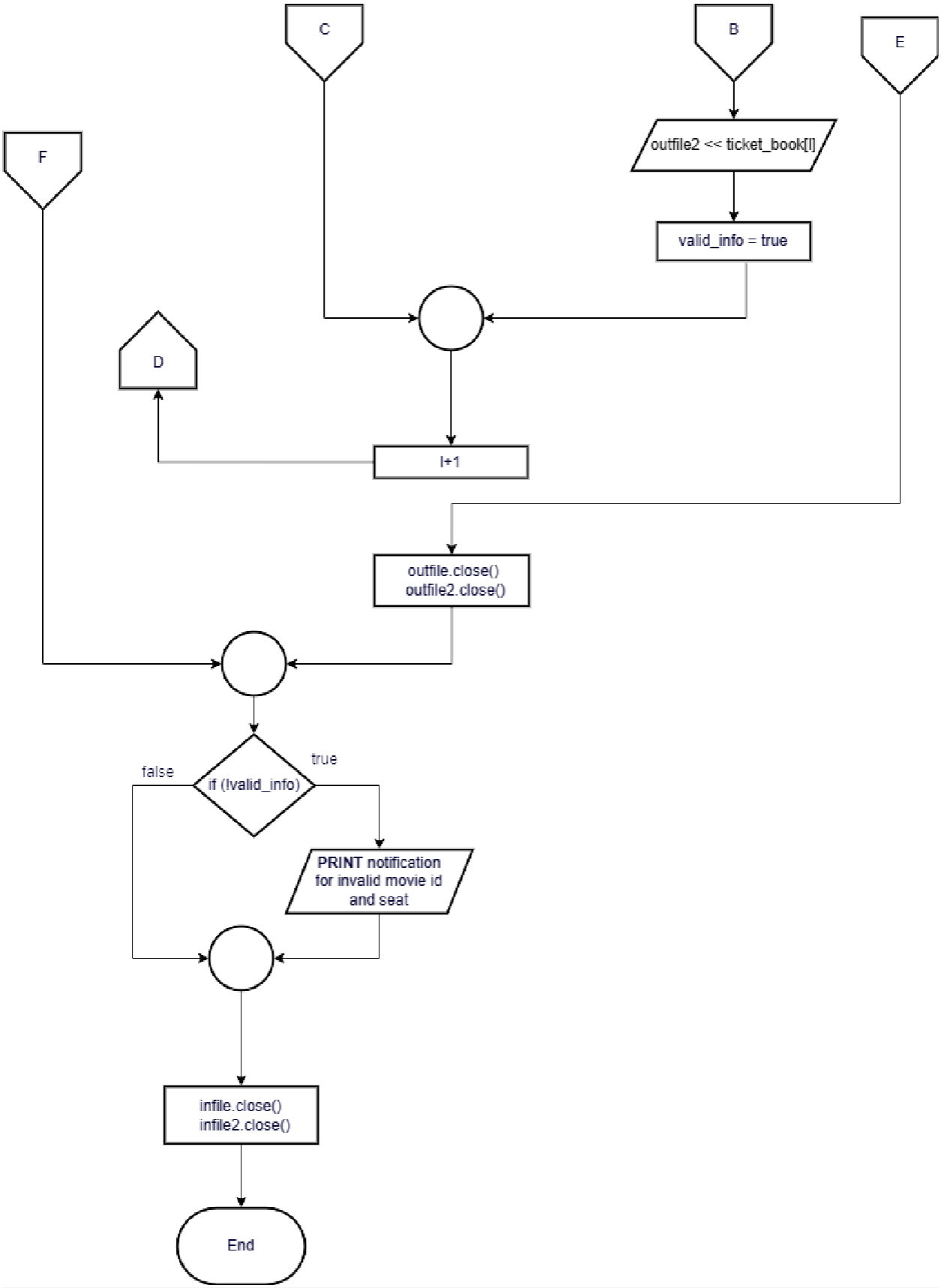
bool edit_cart()



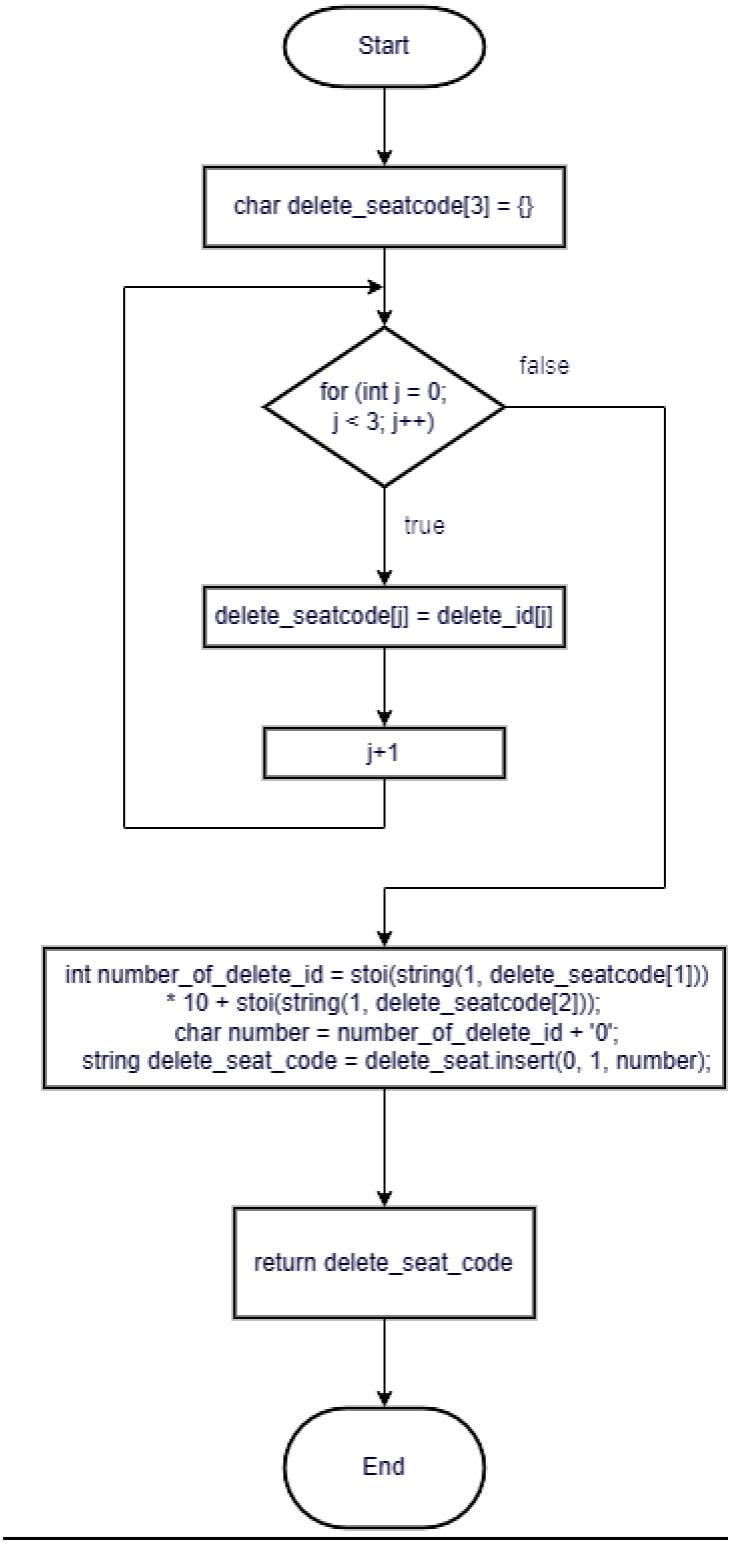
void delete_id_seat(string delete_id, string delete_seat)

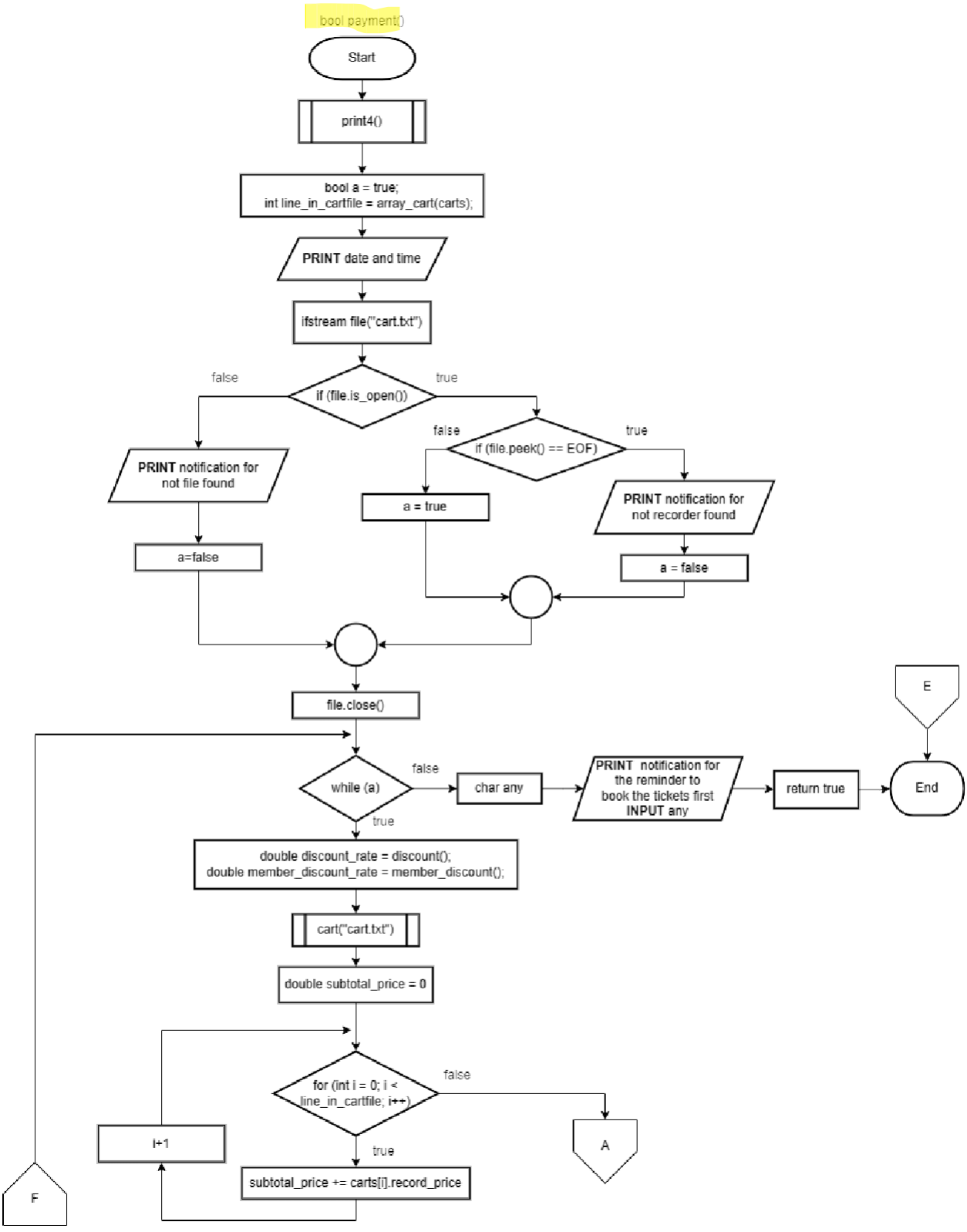


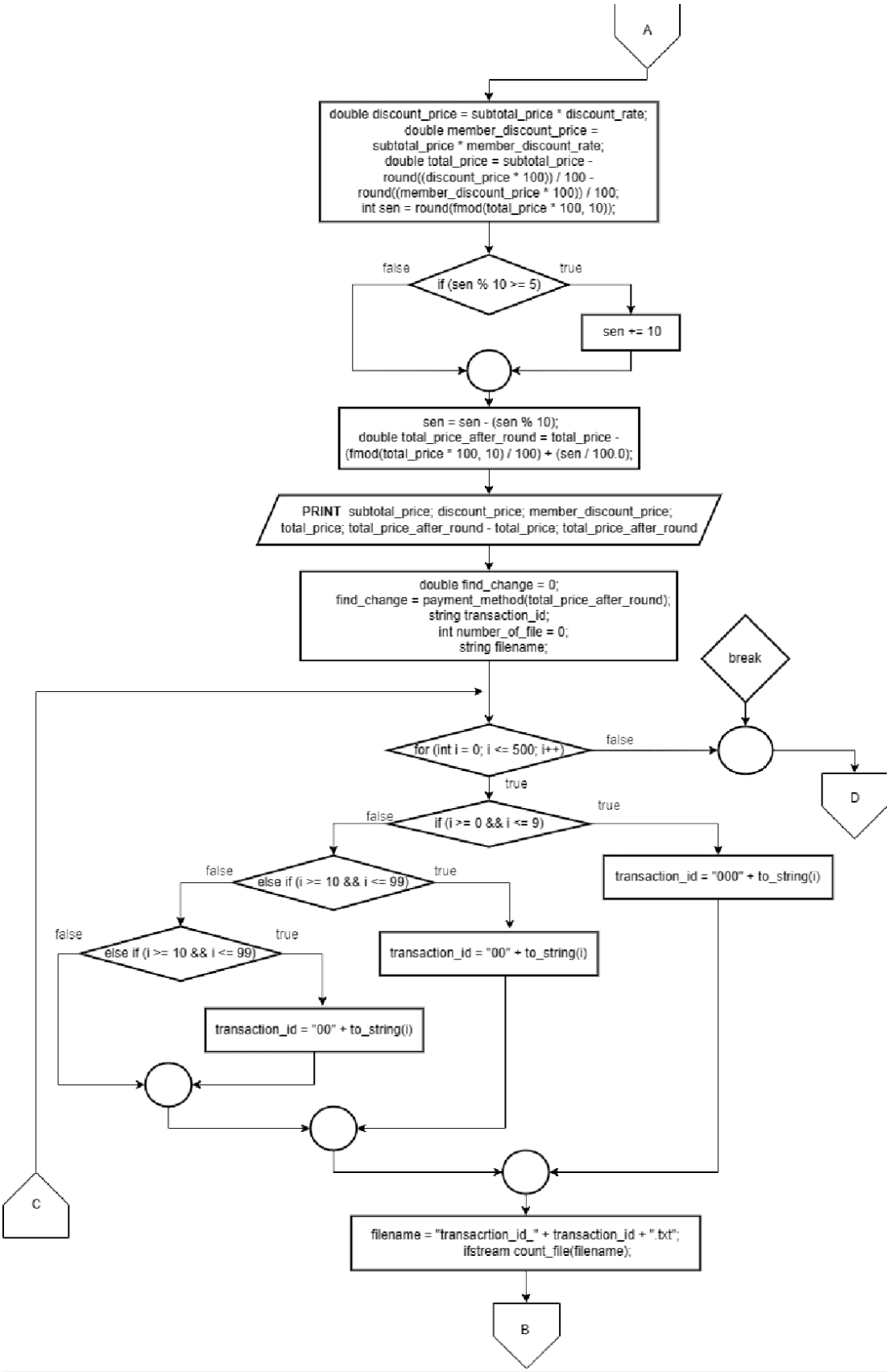

A
if (isalpha(delete_id[0]) && isdigit(delete_id[1]) && isdigit(delete_id[2]) && isalpha(delete_seat[0]) && isdigit(delete_seat[1]) && isdigit(delete_id[2]) && delete_id.length() == 3 && delete_seat.length() == 3)
true
F
ofstream outfile("cart.txt")
ofstream outfile2("booking.txt")
for (int k = 0; k < num_book_in_cart; k++)
false
true
if (infile.is_open() && carts[k].record_id.find(delete_id, 0) == string::npos || carts[k].record_seat.find(delete_seat, 0) == string::npos)
false
true
delete_seat_code = delete_id_seat_code (delete_seat, delete_id)
outfile << cart_array[k]
k+1
for (int l = 0; l < count_line; l++)
false
true
if (infile2.is_open() && ticket_book[l].find(delete_seat_code, 0) == string::npos)
D
C
B
E

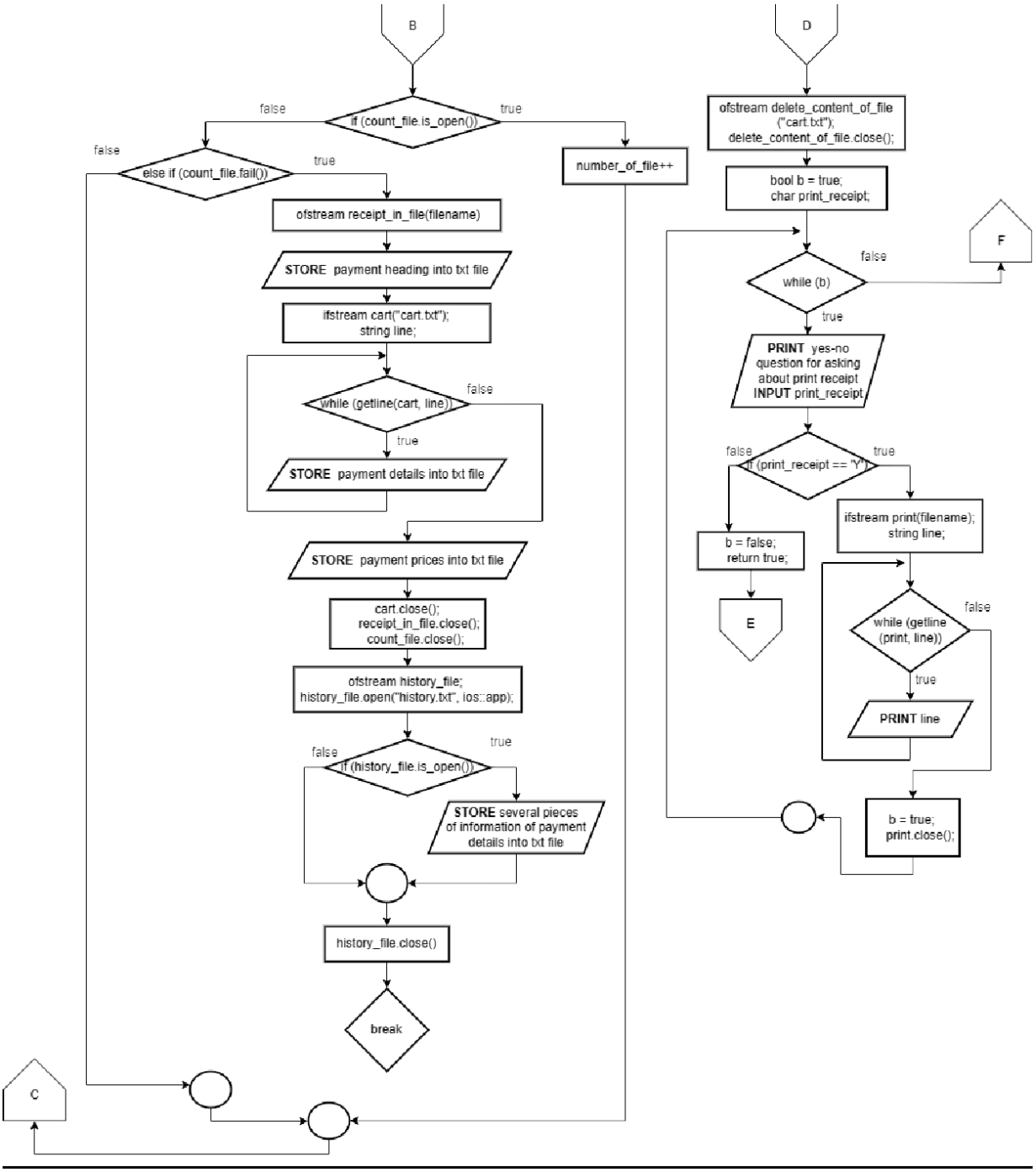

C
B
E
F
outfile2 << ticket_book[I]
valid_info = true
D
I+1
outfile.close()
outfile2.close()
false
if (!valid_info)
true
PRINT notification for invalid movie id and seat
infile.close()
infile2.close()
End

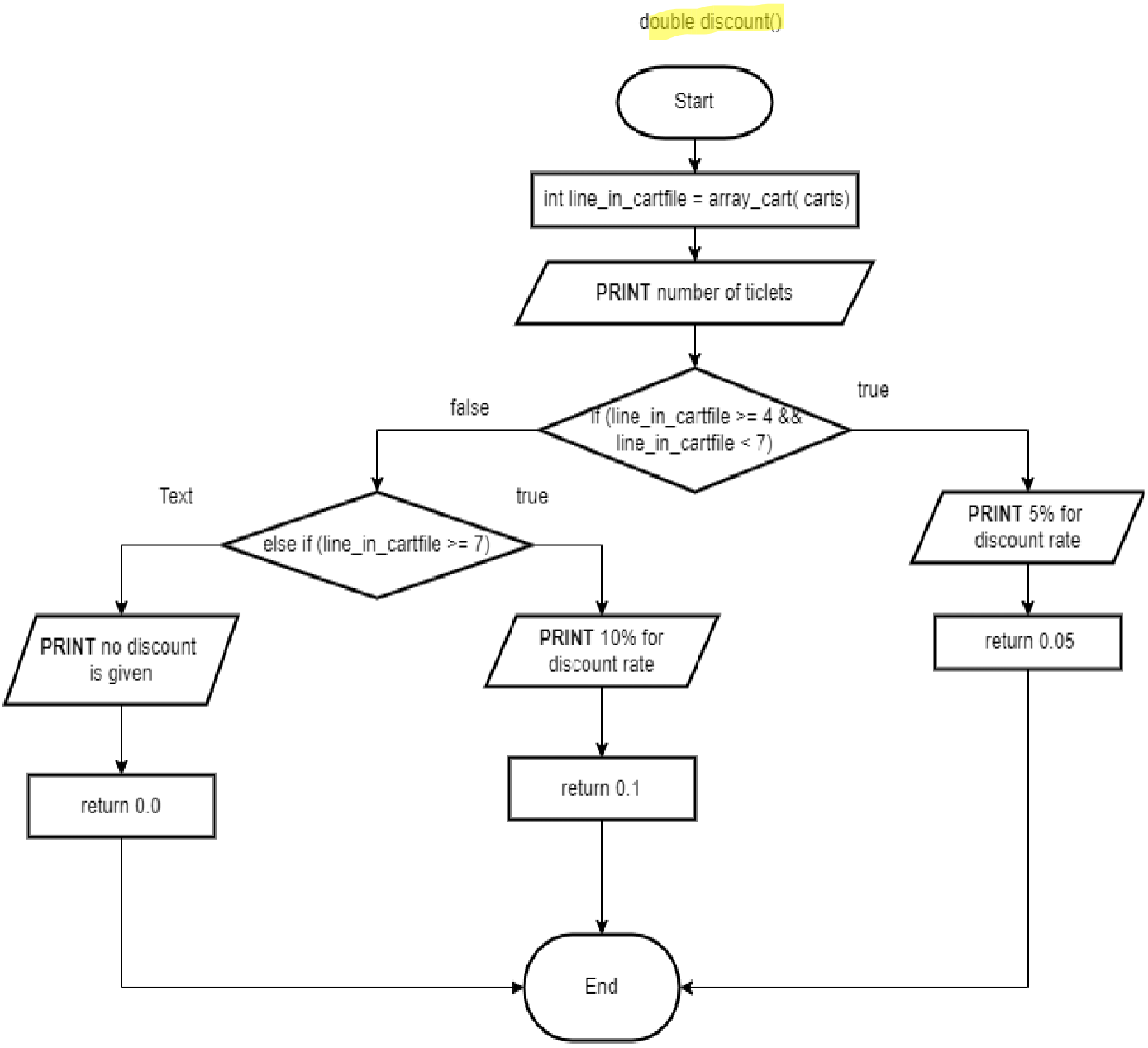
string delete_id_seat_code(string delete_seat, string delete_id)

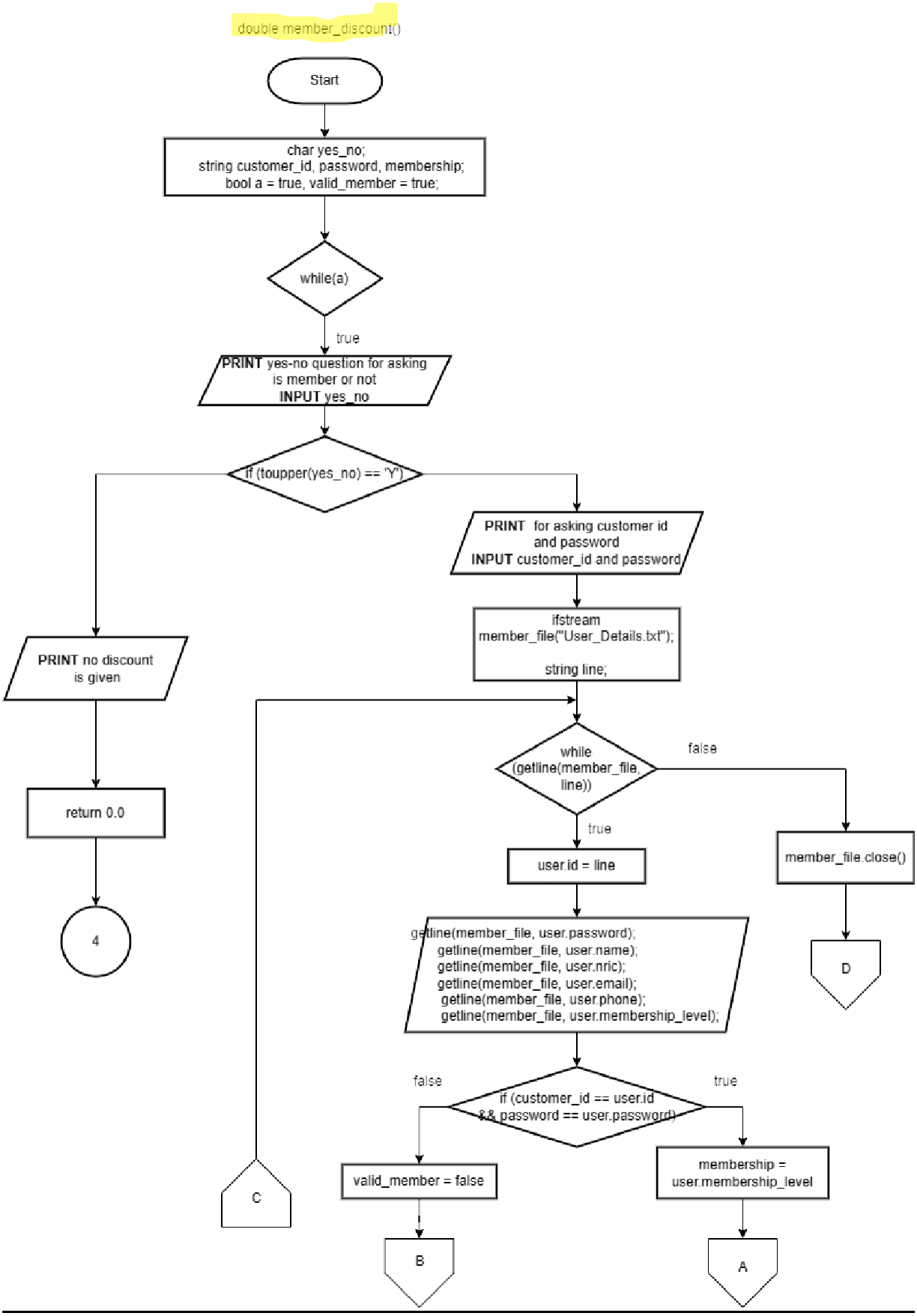


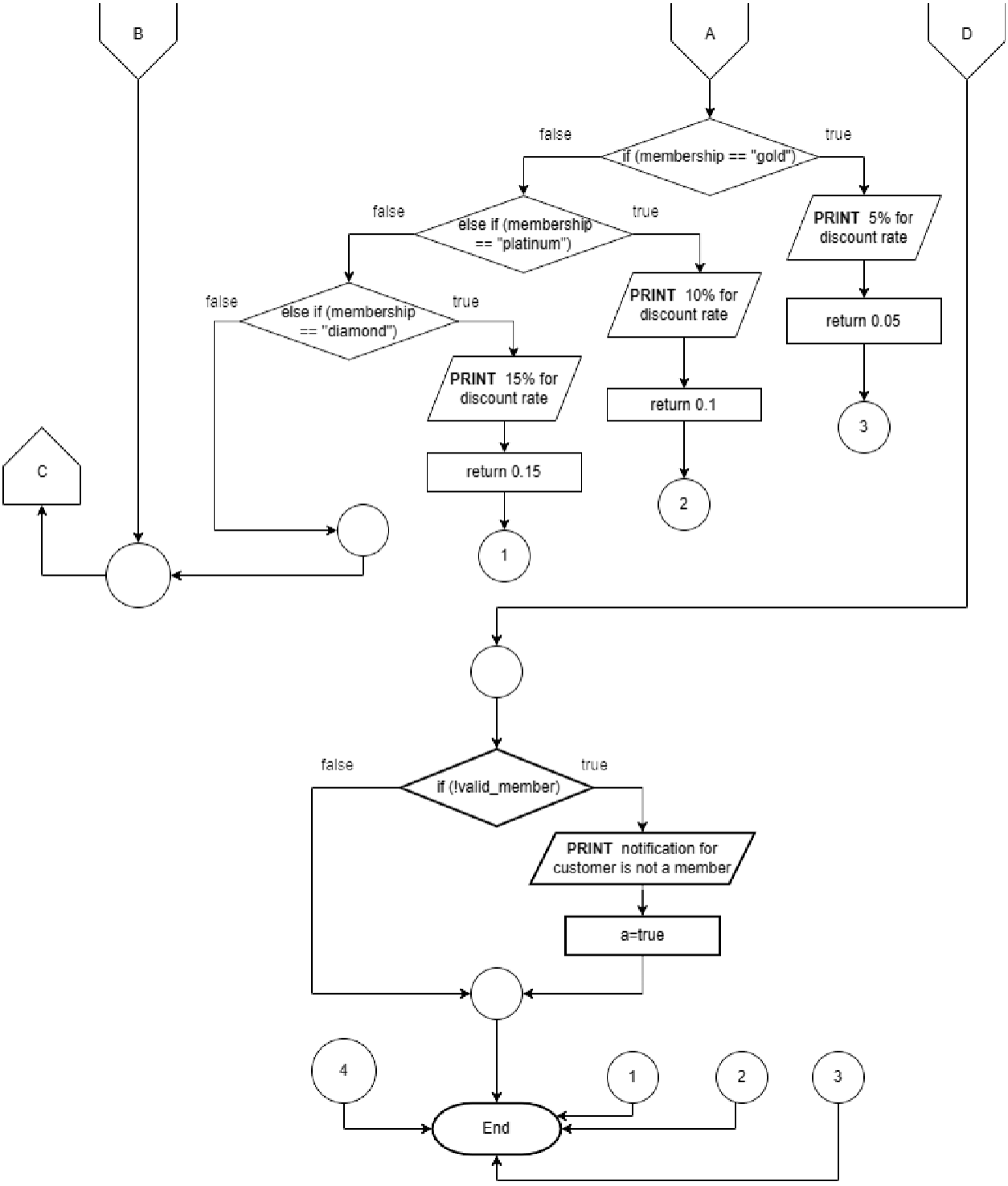

bool payment()
Start
print4()
bool a = true;
int line_in_cartfile = array_cart(carts);
PRINT date and time
ifstream file("cart.txt")
false
if (file.is_open())
true
PRINT notification for not file found
a=false
false
if (file.peek() == EOF)
true
a = true
PRINT notification for not recorder found
a = false
file.close()
while (a)
false
char any
PRINT notification for the reminder to book the tickets first INPUT any
return true
End
E
true
double discount_rate = discount();
double member_discount_rate = member_discount();
cart("cart.txt")
double subtotal_price = 0
for (int i = 0; i < line_in_cartfile; i++)
false
A
true
subtotal_price += carts[i].record_price
i+1
F

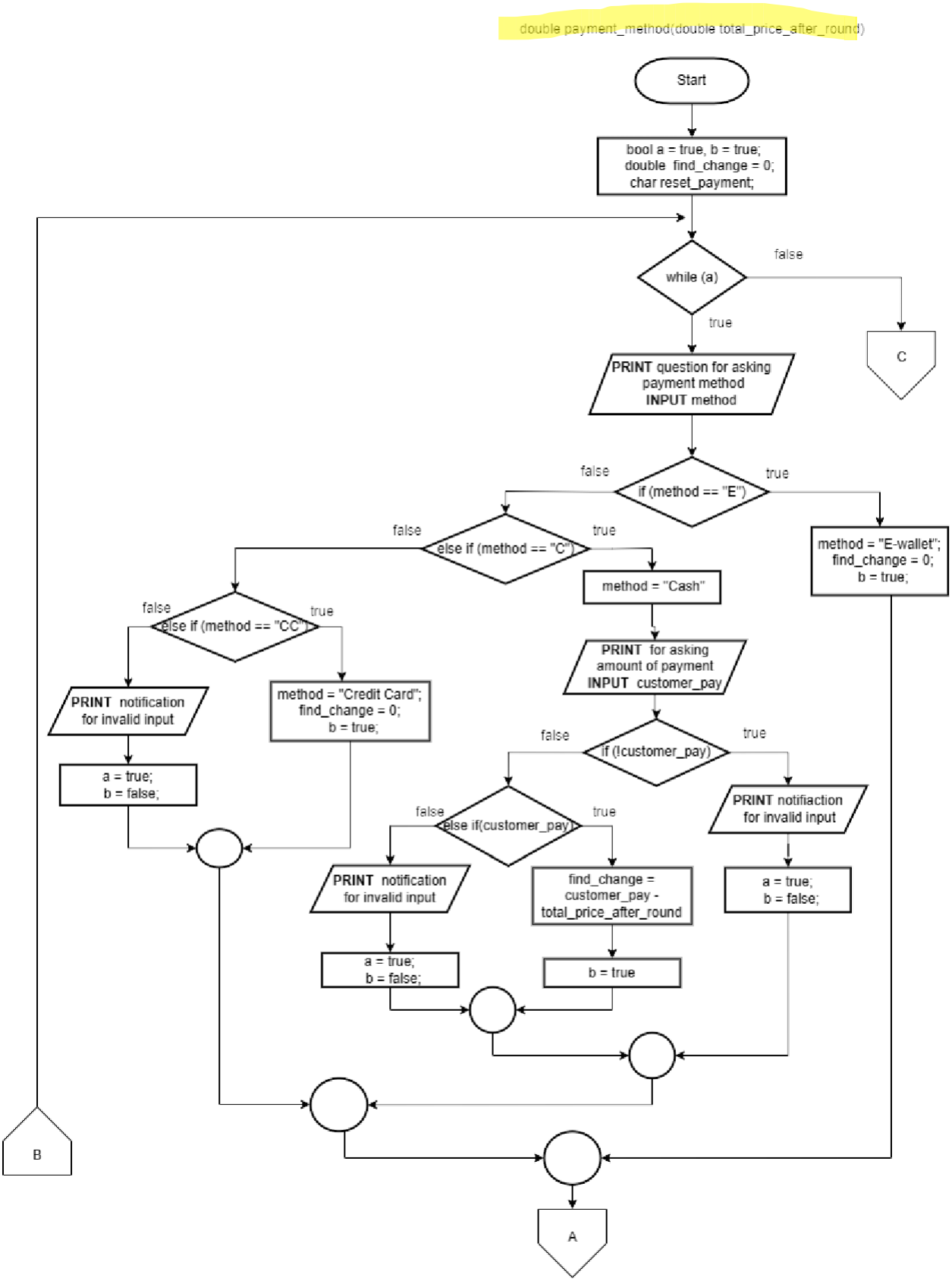

A
double discount_price = subtotal_price * discount_rate;
double member_discount_price =
subtotal_price * member_discount_rate;
double total_price = subtotal_price -
round((discount_price * 100)) / 100 -
round((member_discount_price * 100)) / 100;
int sen = round(fmod(total_price * 100, 10));
false
if (sen % 10 >= 5)
true
sen += 10
sen = sen - (sen % 10);
double total_price_after_round = total_price -
(fmod(total_price * 100, 10) / 100) + (sen / 100.0);
PRINT subtotal_price; discount_price; member_discount_price;
total_price; total_price_after_round - total_price; total_price_after_round
double find_change = 0;
find_change = payment_method(total_price_after_round);
string transaction_id;
int number_of_file = 0;
string filename;
break
for (int i = 0; i <= 500; i++)
false
true
D
false
if (i >= 0 && i <= 9)
true
transaction_id = "000" + to_string(i)
false
else if (i >= 10 && i <= 99)
true
false
else if (i >= 10 && i <= 99)
true
transaction_id = "00" + to_string(i)
transaction_id = "00" + to_string(i)
C
filename = "transacrtion_id_" + transaction_id + ".txt";
ifstream count_file(filename);
B

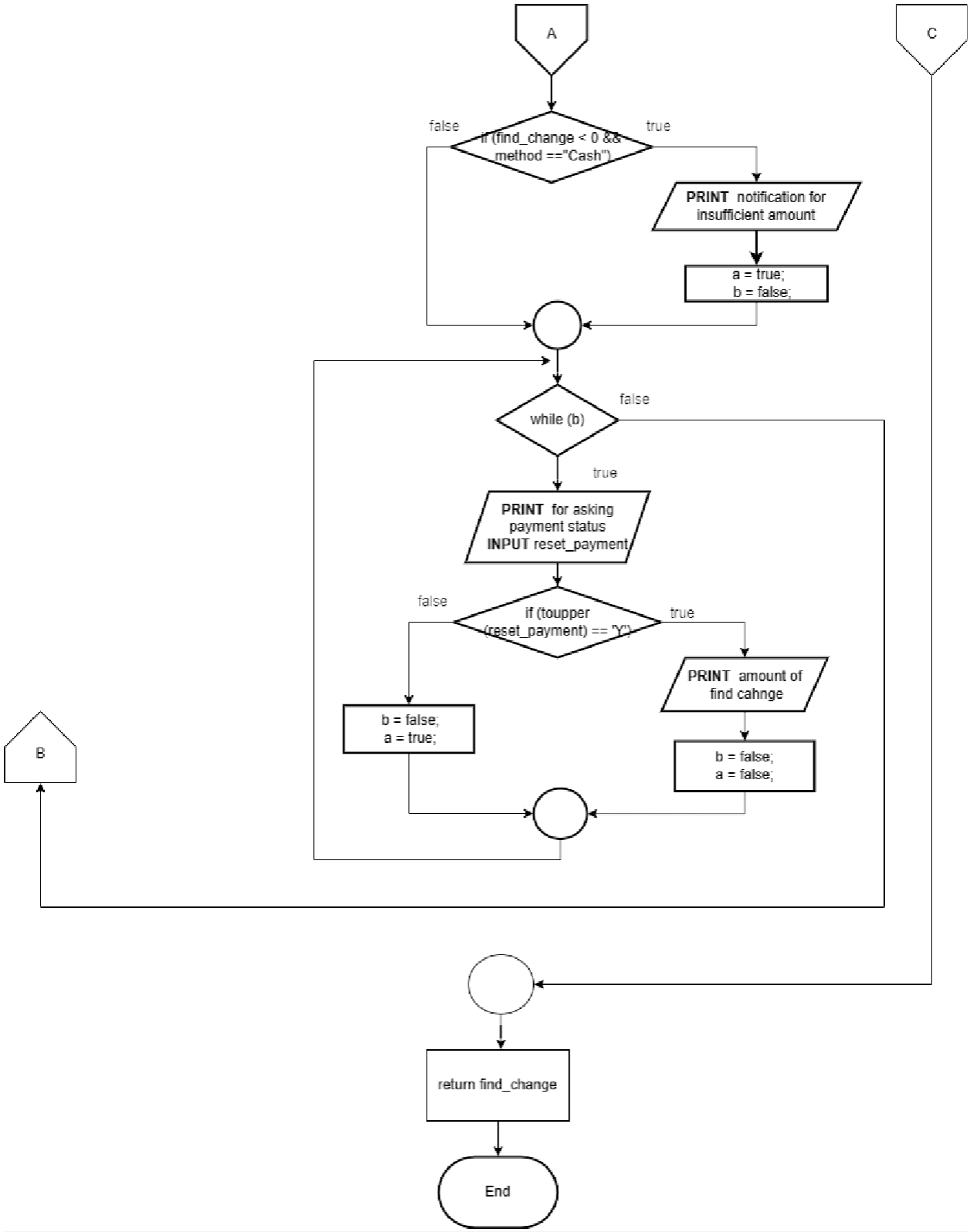

B
if (count_file.is_open())
false
true
else if (count_file.fail())
false
true
number_of_file++
ofstream receipt_in_file(filename)
STORE payment heading into txt file
ifstream cart("cart.txt");
string line;
while (getline(cart, line))
false
true
STORE payment details into txt file
STORE payment prices into txt file
cart.close();
receipt_in_file.close();
count_file.close();
ofstream history_file;
history_file.open("history.txt", ios::app);
if (history_file.is_open())
false
true
STORE several pieces of information of payment details into txt file
history_file.close()
break
C
D
ofstream delete_content_of_file ("cart.txt");
delete_content_of_file.close();
bool b = true;
char print_receipt;
F
while (b)
false
true
PRINT yes-no question for asking about print receipt
INPUT print_receipt
if (print_receipt == 'Y')
false
true
b = false;
return true;
E
ifstream print(filename);
string line;
while (getline (print, line))
false
true
PRINT line
b = true;
print.close();

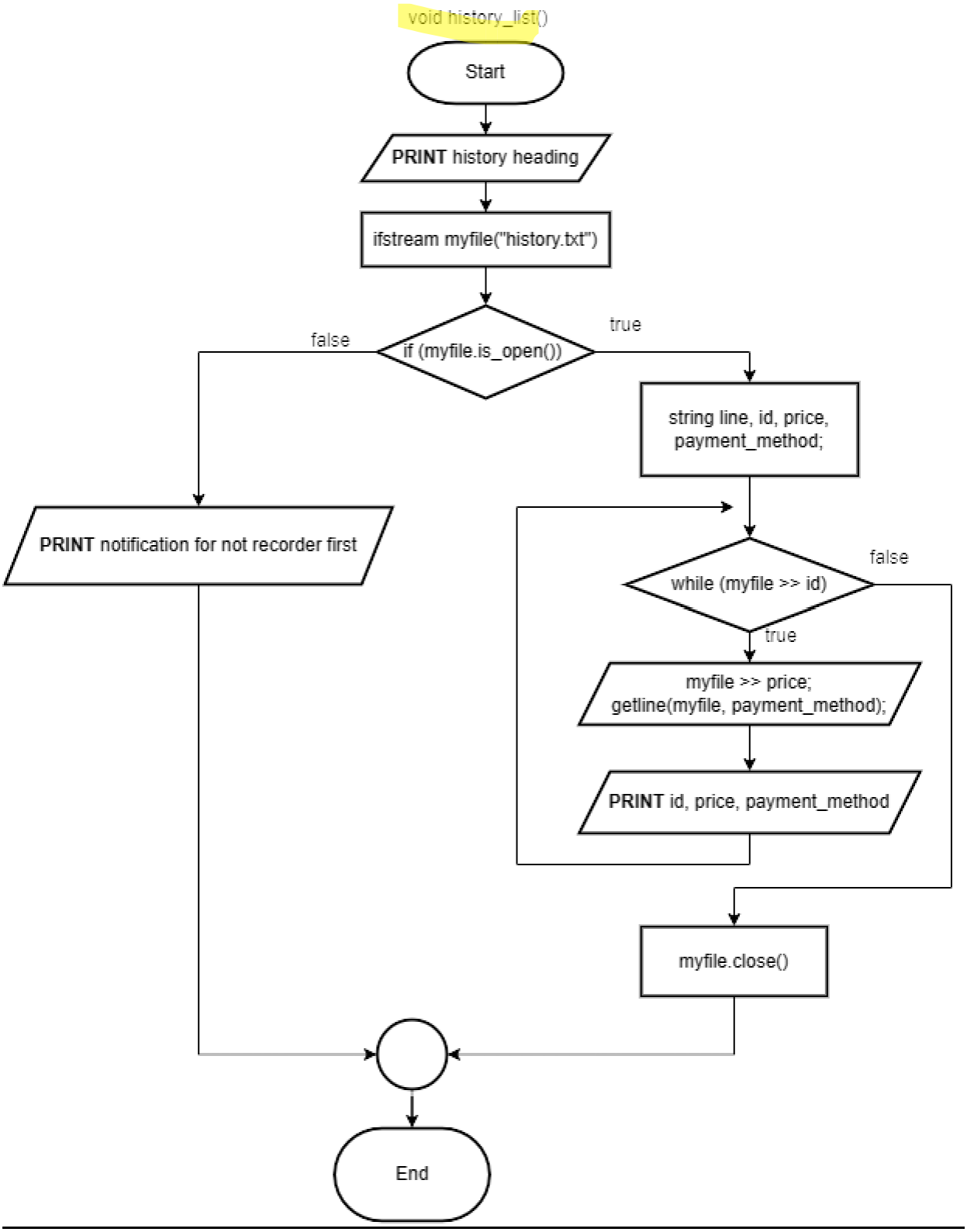

double discount()
Start
int line_in_cartfile = array_cart(carts)
PRINT number of ticlets
false
if (line_in_cartfile >= 4 && line_in_cartfile < 7)
true
Text
true
else if (line_in_cartfile >= 7)
PRINT no discount is given
PRINT 10% for discount rate
PRINT 5% for discount rate
return 0.0
return 0.1
return 0.05
End

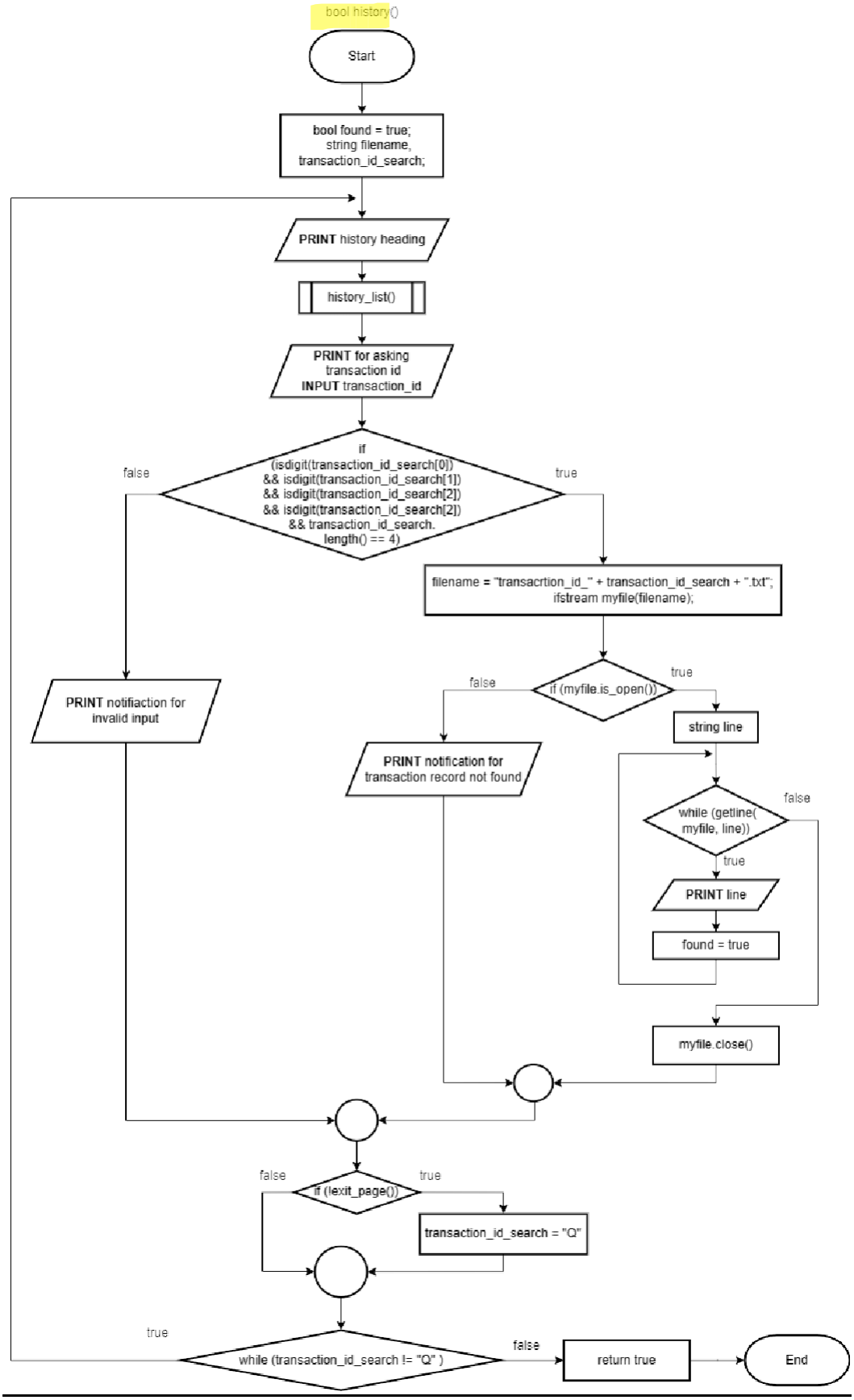

double member_discount()
Start
char yes_no;
string customer_id, password, membership;
bool a = true, valid_member = true;
while(a)
true
PRINT yes-no question for asking
is member or not
INPUT yes_no
if (toupper(yes_no) == 'Y')
PRINT no discount
is given
return 0.0
4
PRINT for asking customer id
and password
INPUT customer_id and password
ifstream
member_file("User_Details.txt");
string line;
while
(getline(member_file,
line))
false
true
user.id = line
member_file.close()
D
getline(member_file, user.password);
getline(member_file, user.name);
getline(member_file, user.nric);
getline(member_file, user.email);
getline(member_file, user.phone);
getline(member_file, user.membership_level);
false
if (customer_id == user.id
&& password == user.password)
true
valid_member = false
membership =
user.membership_level
C
B
A

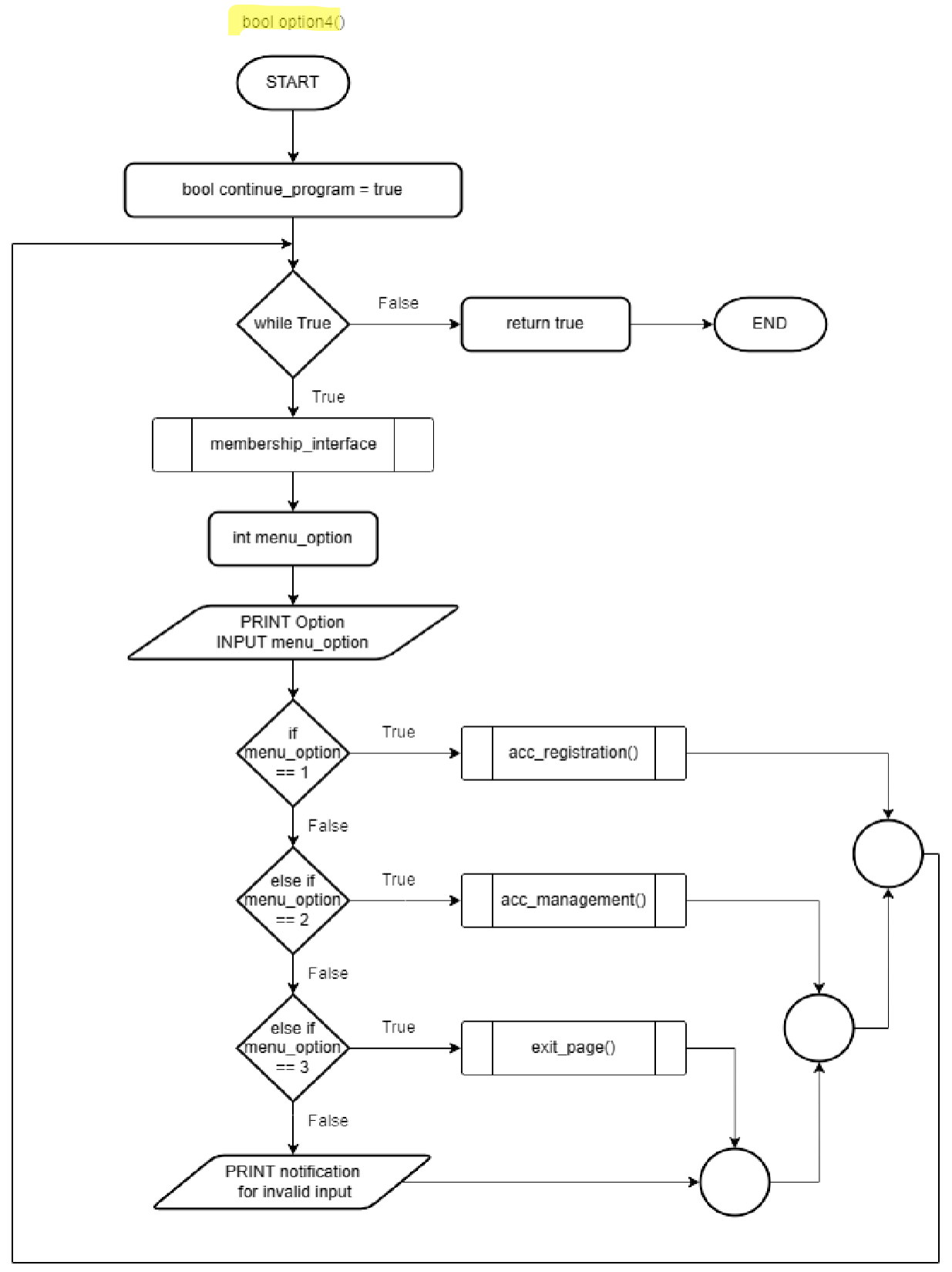

B
A
D
false
if (membership == "gold")
true
false
else if (membership == "platinum")
true
PRINT 5% for discount rate
return 0.05
3
false
else if (membership == "diamond")
true
PRINT 10% for discount rate
return 0.1
2
PRINT 15% for discount rate
return 0.15
1
C
false
if (!valid_member)
true
PRINT notification for customer is not a member
a=true
4
1
2
3
End

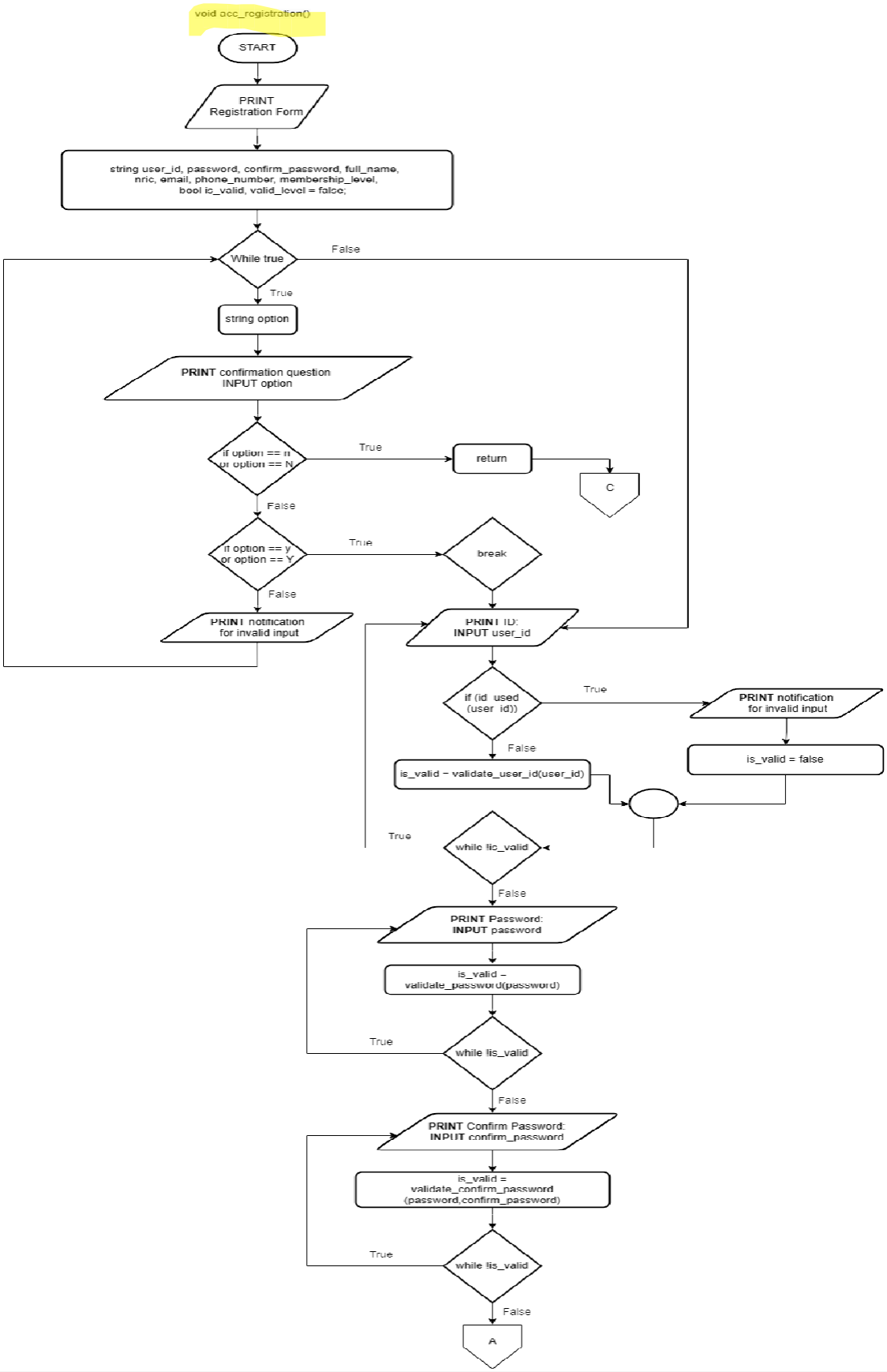

double payment_method(double total_price_after_round)
Start
bool a = true, b = true;
double find_change = 0;
char reset_payment;
while (a)
false
true
C
PRINT question for asking payment method
INPUT method
if (method == "E")
false
true
else if (method == "C")
false
true
method = "E-wallet";
find_change = 0;
b = true;
method = "Cash"
else if (method == "CC")
false
true
PRINT for asking amount of payment
INPUT customer_pay
PRINT notification for invalid input
method = "Credit Card";
find_change = 0;
b = true;
if (!customer_pay)
false
true
a = true;
b = false;
else if(customer_pay)
false
true
PRINT notifiaction for invalid input
PRINT notification for invalid input
find_change = customer_pay - total_price_after_round
a = true;
b = false;
b = true
B
A

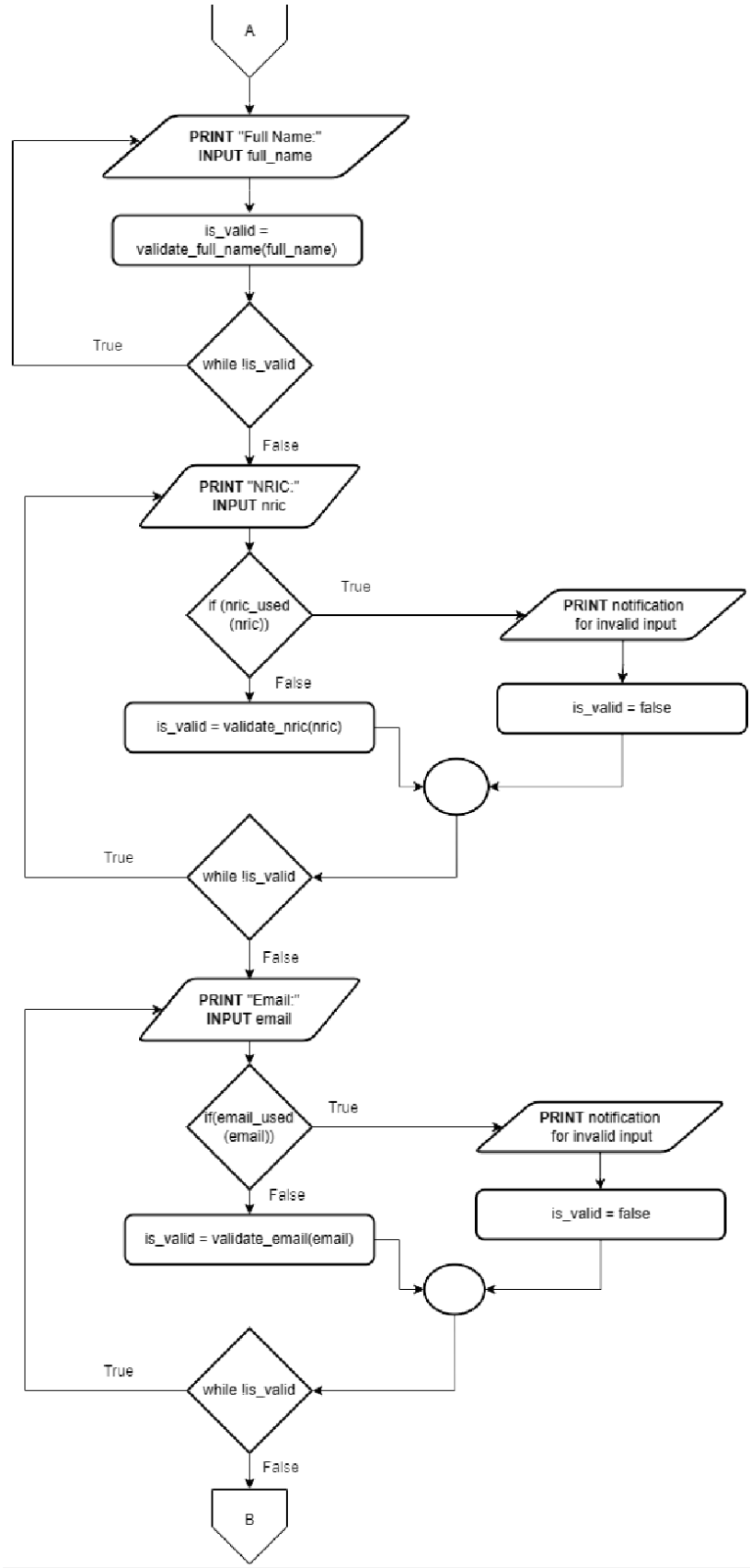

A
C
false
if (find_change < 0 &&
method =="Cash")
true
PRINT notification for
insufficient amount
a = true;
b = false;
while (b)
false
true
PRINT for asking
payment status
INPUT reset_payment
false
if (toupper
(reset_payment) == 'Y')
true
PRINT amount of
find cahnge
b = false;
a = true;
b = false;
a = false;
B
return find_change
End

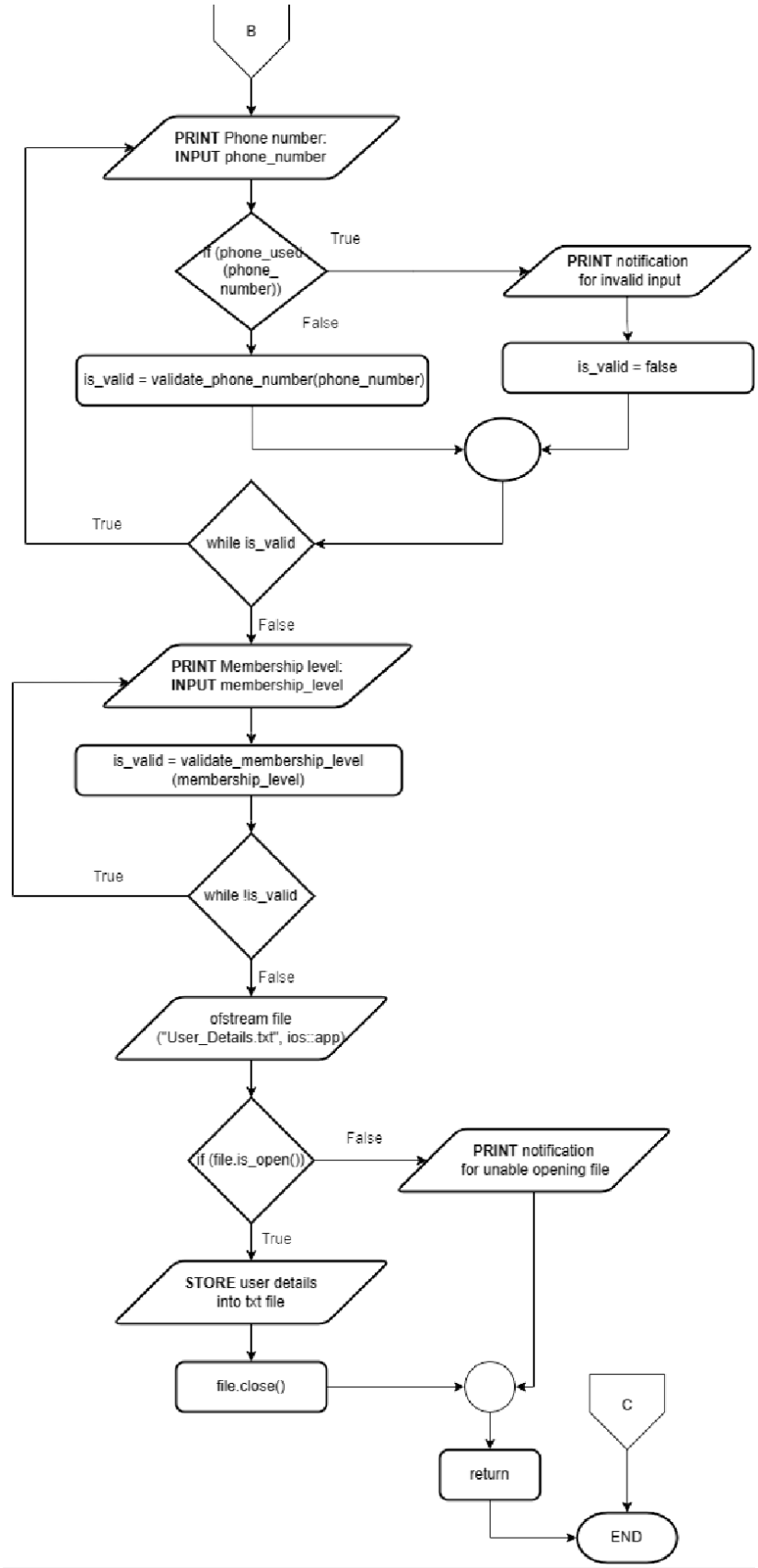

void history_list()
Start
PRINT history heading
ifstream myfile("history.txt")
if (myfile.is_open())
false
true
PRINT notification for not recorder first
string line, id, price, payment_method;
while (myfile >> id)
false
true
myfile >> price;
getline(myfile, payment_method);
PRINT id, price, payment_method
myfile.close()
End

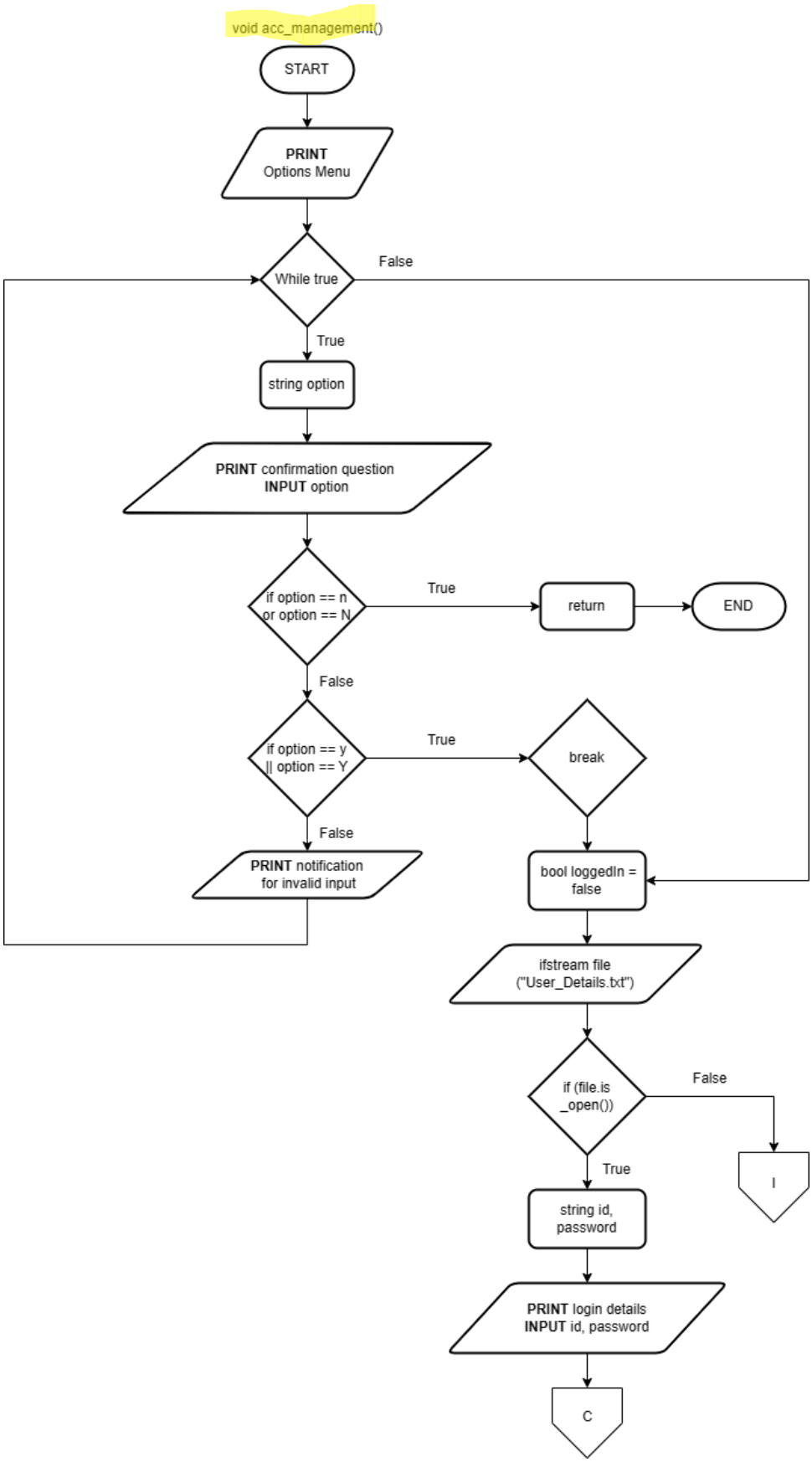

bool history()
Start
bool found = true;
string filename,
transaction_id_search;
PRINT history heading
history_list()
PRINT for asking
transaction id
INPUT transaction_id
if
(isdigit(transaction_id_search[0])
&& isdigit(transaction_id_search[1])
&& isdigit(transaction_id_search[2])
&& isdigit(transaction_id_search[2])
&& transaction_id_search.
length() == 4)
false
true
PRINT notifiaction for
invalid input
filename = "transacrtion_id_" + transaction_id_search + ".txt";
ifstream myfile(filename);
if (myfile.is_open())
false
true
PRINT notification for
transaction record not found
string line
while (getline(
myfile, line))
false
true
PRINT line
found = true
myfile.close()
if (!exit_page())
false
true
transaction_id_search = "Q"
while (transaction_id_search != "Q")
true
false
return true
End

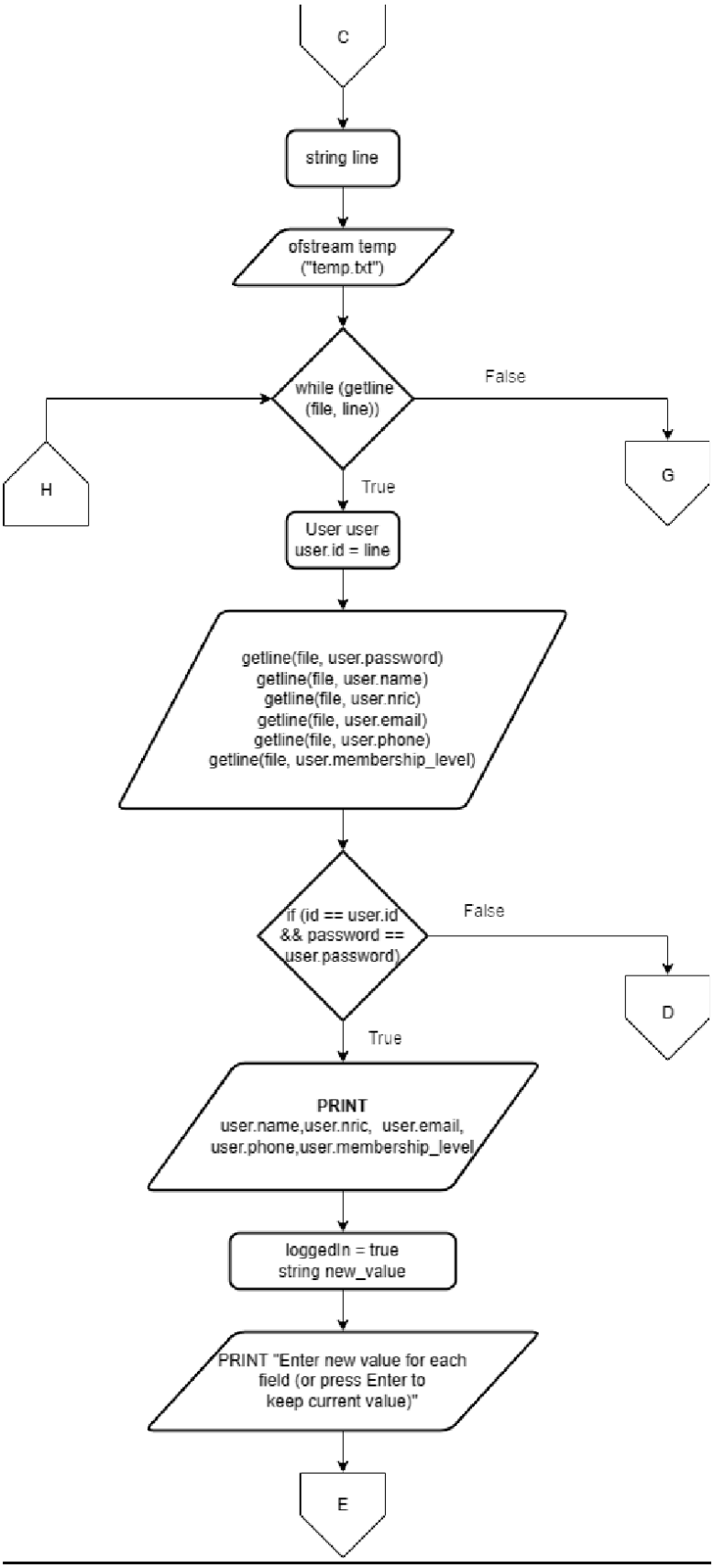

bool option4()
START
bool continue_program = true
while True
False
return true
END
True
membership_interface
int menu_option
PRINT Option
INPUT menu_option
if
menu_option
== 1
True
acc_registration()
False
else if
menu_option
== 2
True
acc_management()
False
else if
menu_option
== 3
True
exit_page()
False
PRINT notification
for invalid input

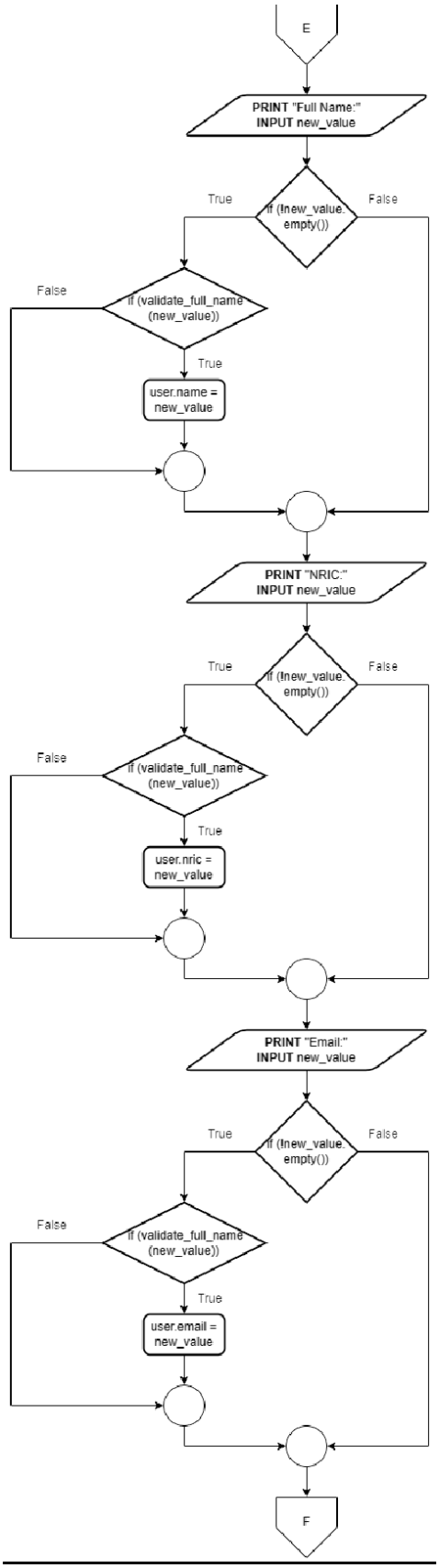

void acc_registration()
START
PRINT
Registration Form
string user_id, password, confirm_password, full_name,
nric, email, phone_number, membership_level,
bool is_valid, valid_level = false;
While true
False
True
string option
PRINT confirmation question
INPUT option
if option == n
or option == N
True
return
C
False
if option == y
or option == Y
True
break
False
PRINT notification
for invalid input
PRINT ID:
INPUT user_id
if (id_used
(user_id))
True
PRINT notification
for invalid input
is_valid = false
False
is_valid = validate_user_id(user_id)
True
while !is_valid
False
PRINT Password:
INPUT password
is_valid =
validate_password(password)
True
while !is_valid
False
PRINT Confirm Password:
INPUT confirm_password
is_valid =
validate_confirm_password
(password,confirm_password)
True
while !is_valid
False
A

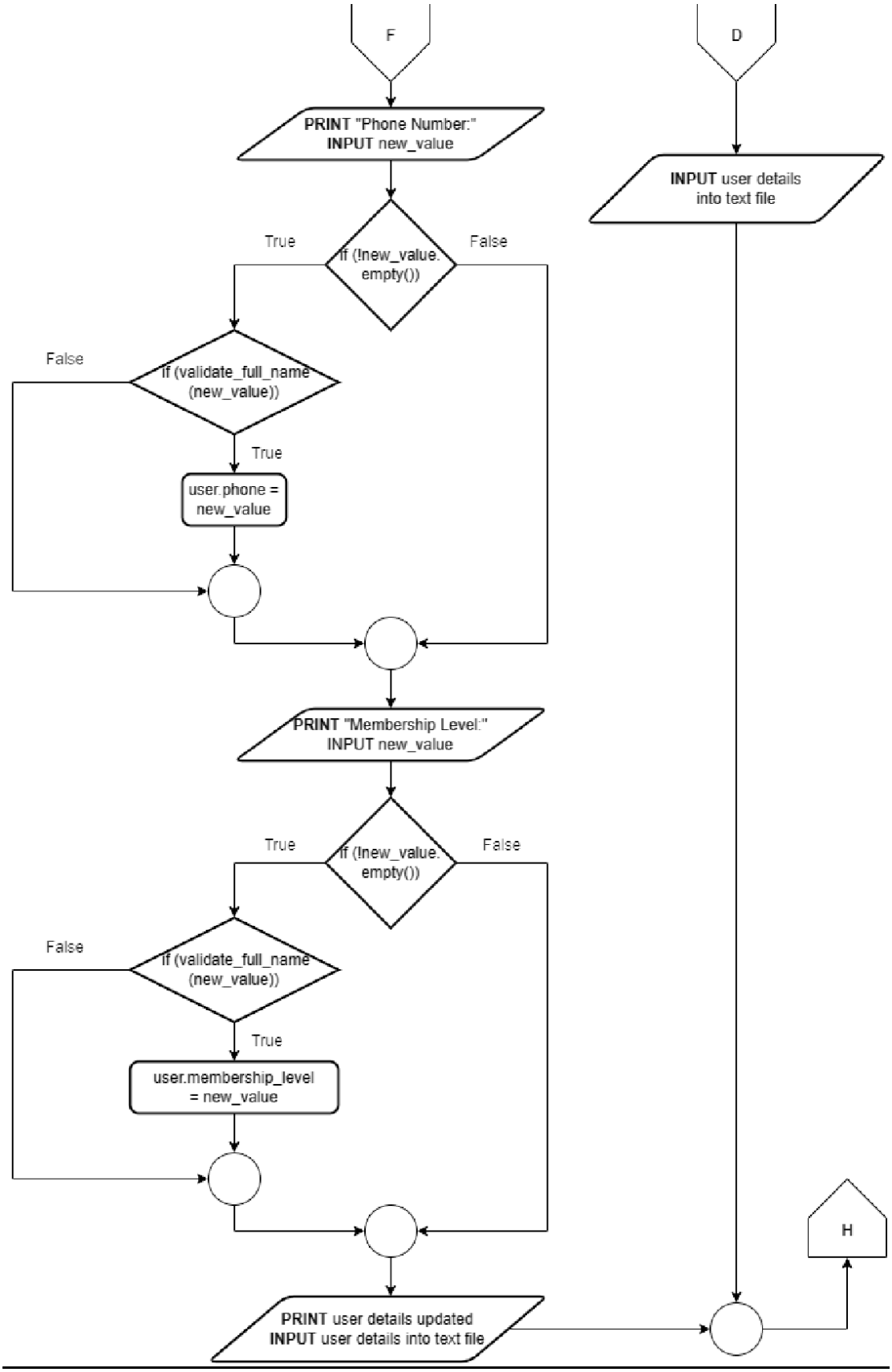

A
PRINT "Full Name:"
INPUT full_name
is_valid = validate_full_name(full_name)
while !is_valid
True
False
PRINT "NRIC:"
INPUT nric
if (nric_used (nric))
True
PRINT notification for invalid input
is_valid = false
False
is_valid = validate_nric(nric)
while !is_valid
True
False
PRINT "Email:"
INPUT email
if(email_used (email))
True
PRINT notification for invalid input
is_valid = false
False
is_valid = validate_email(email)
while !is_valid
True
False
B

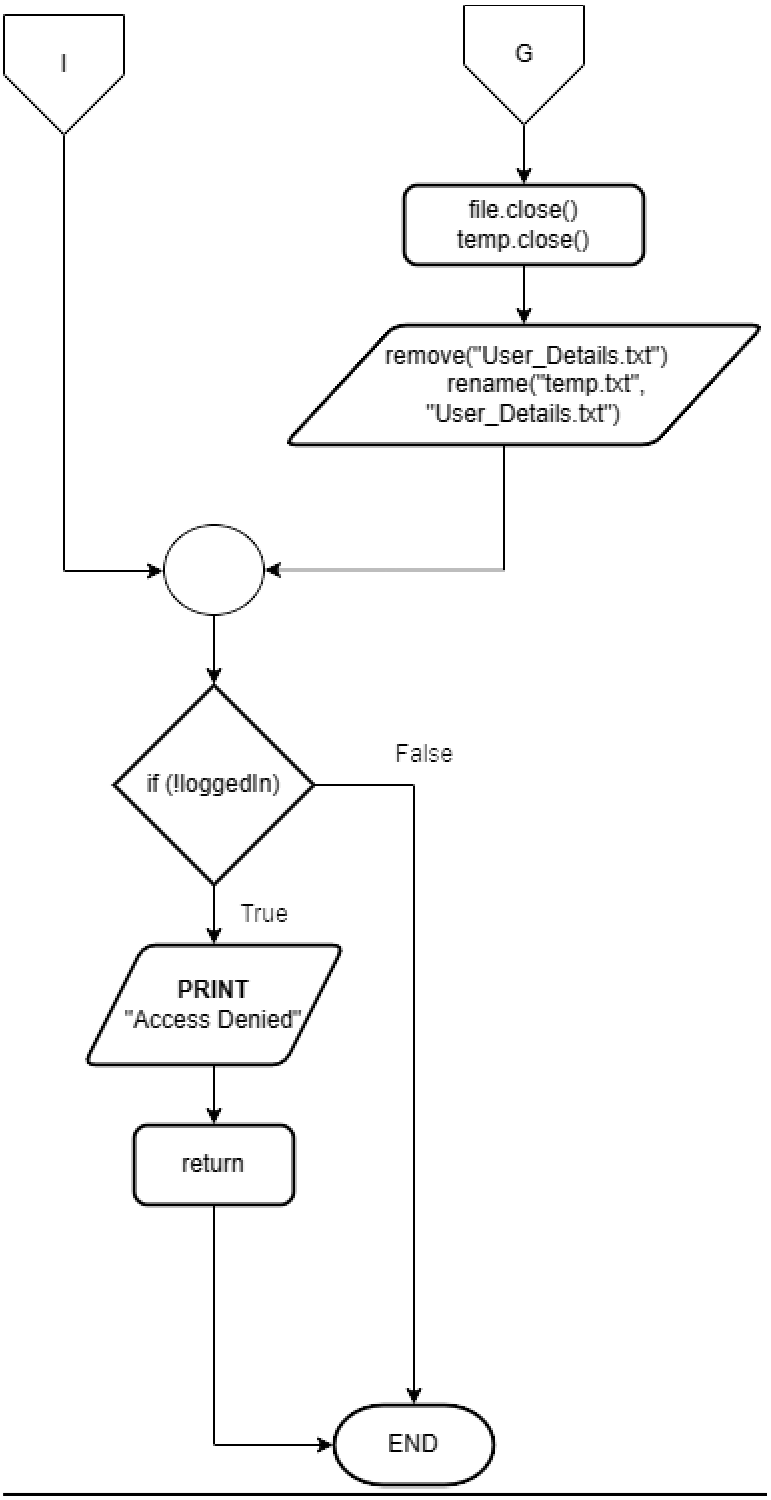

B
PRINT Phone number:
INPUT phone_number
if (phone_used (phone_ number))
True
PRINT notification for invalid input
False
is_valid = validate_phone_number(phone_number)
is_valid = false
True
while is_valid
False
PRINT Membership level:
INPUT membership_level
is_valid = validate_membership_level (membership_level)
True
while !is_valid
False
ofstream file ("User_Details.txt", ios::app)
if (file.is_open())
False
PRINT notification for unable opening file
True
STORE user details into txt file
file.close()
C
return
END


void acc_management()
START
PRINT
Options Menu
While true
False
True
string option
PRINT confirmation question
INPUT option
if option == n
or option == N
True
return
END
False
if option == y
|| option == Y
True
break
False
PRINT notification
for invalid input
bool loggedIn =
false
ifstream file
("User_Details.txt")
if (file.is
_open())
False
I
True
string id,
password
PRINT login details
INPUT id, password
C

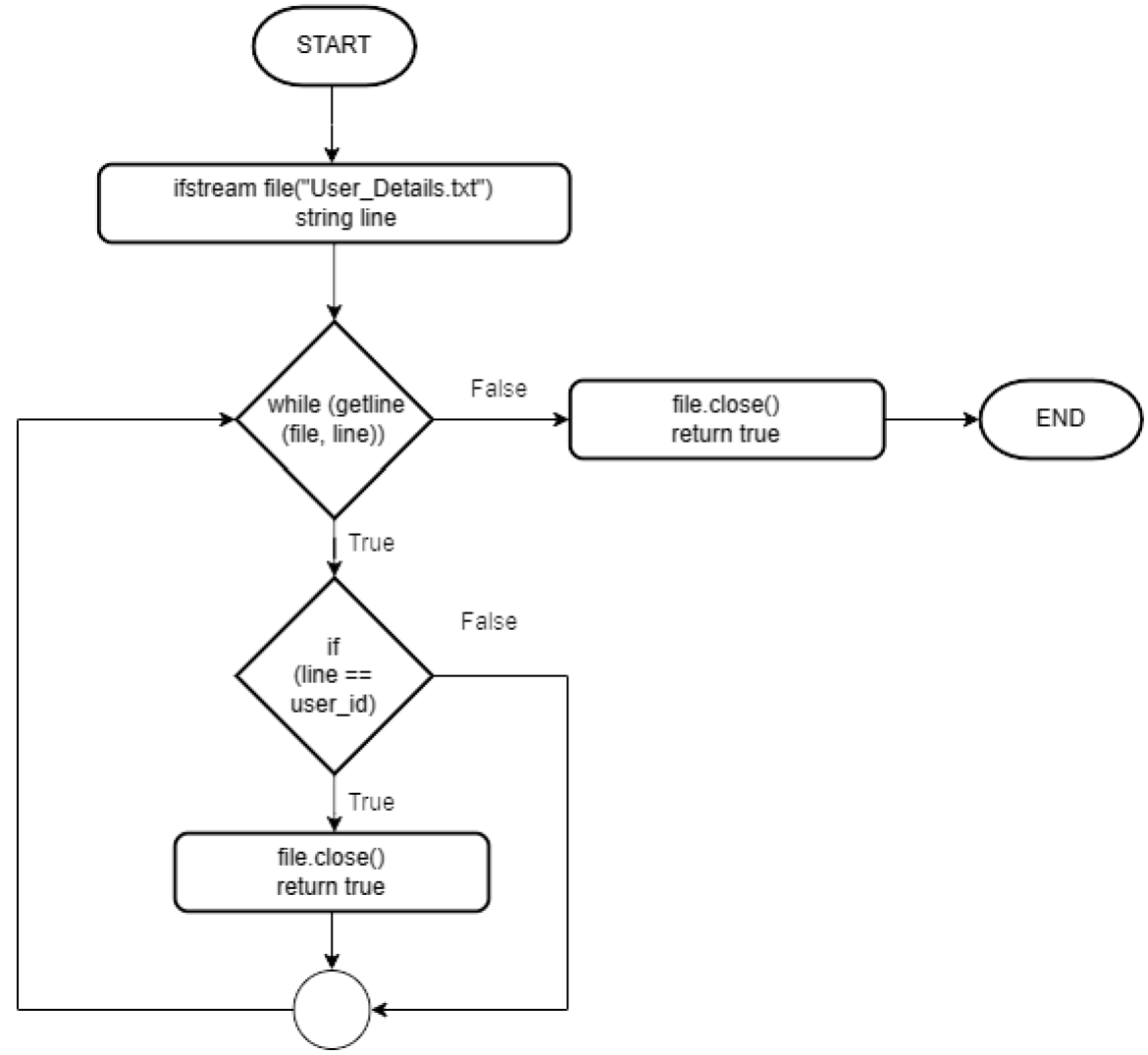

C
string line
ofstream temp ("temp.txt")
while (getline (file, line))
False
H
G
True
User user
user.id = line
getline(file, user.password)
getline(file, user.name)
getline(file, user.nric)
getline(file, user.email)
getline(file, user.phone)
getline(file, user.membership_level)
if (id == user.id && password == user.password)
False
D
True
PRINT
user.name,user.nric, user.email, user.phone,user.membership_level
loggedIn = true
string new_value
PRINT "Enter new value for each field (or press Enter to keep current value)"
E


E
PRINT "Full Name:"
INPUT new_value
True
if (!new_value.empty())
False
False
if (validate_full_name (new_value))
True
user.name = new_value
PRINT "NRIC:"
INPUT new_value
True
if (!new_value.empty())
False
False
if (validate_full_name (new_value))
True
user.nric = new_value
PRINT "Email:"
INPUT new_value
True
if (!new_value.empty())
False
False
if (validate_full_name (new_value))
True
user.email = new_value
F

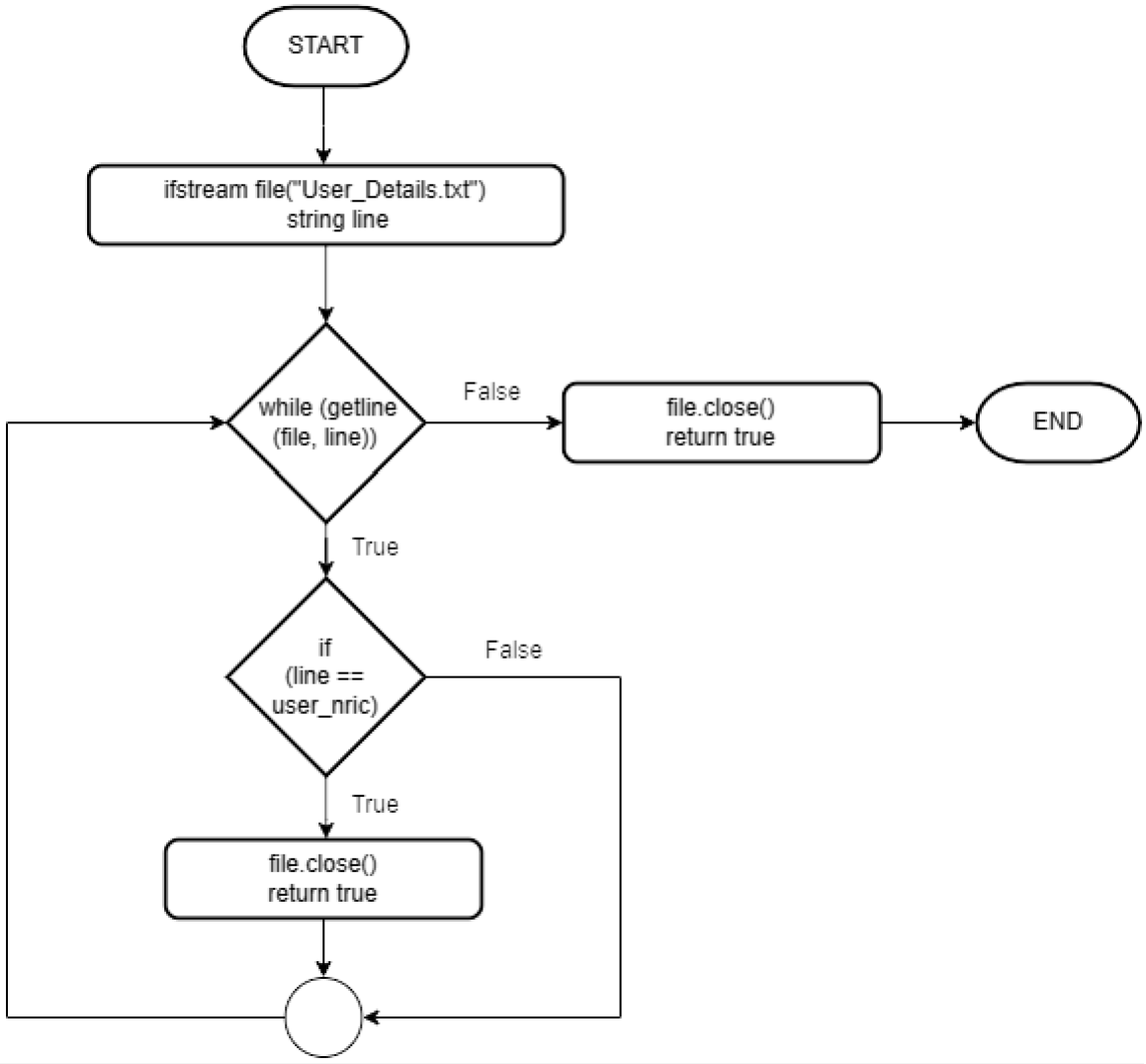

F
D
PRINT "Phone Number:"
INPUT new_value
INPUT user details
into text file
True
if (!new_value.
empty())
False
False
if (validate_full_name
(new_value))
True
user.phone =
new_value
PRINT "Membership Level:"
INPUT new_value
True
if (!new_value.
empty())
False
False
if (validate_full_name
(new_value))
True
user.membership_level
= new_value
H
PRINT user details updated
INPUT user details into text file


I
G
file.close()
temp.close()
remove("User_Details.txt")
rename("temp.txt",
"User_Details.txt")
if (!loggedIn)
False
True
PRINT
"Access Denied"
return
END

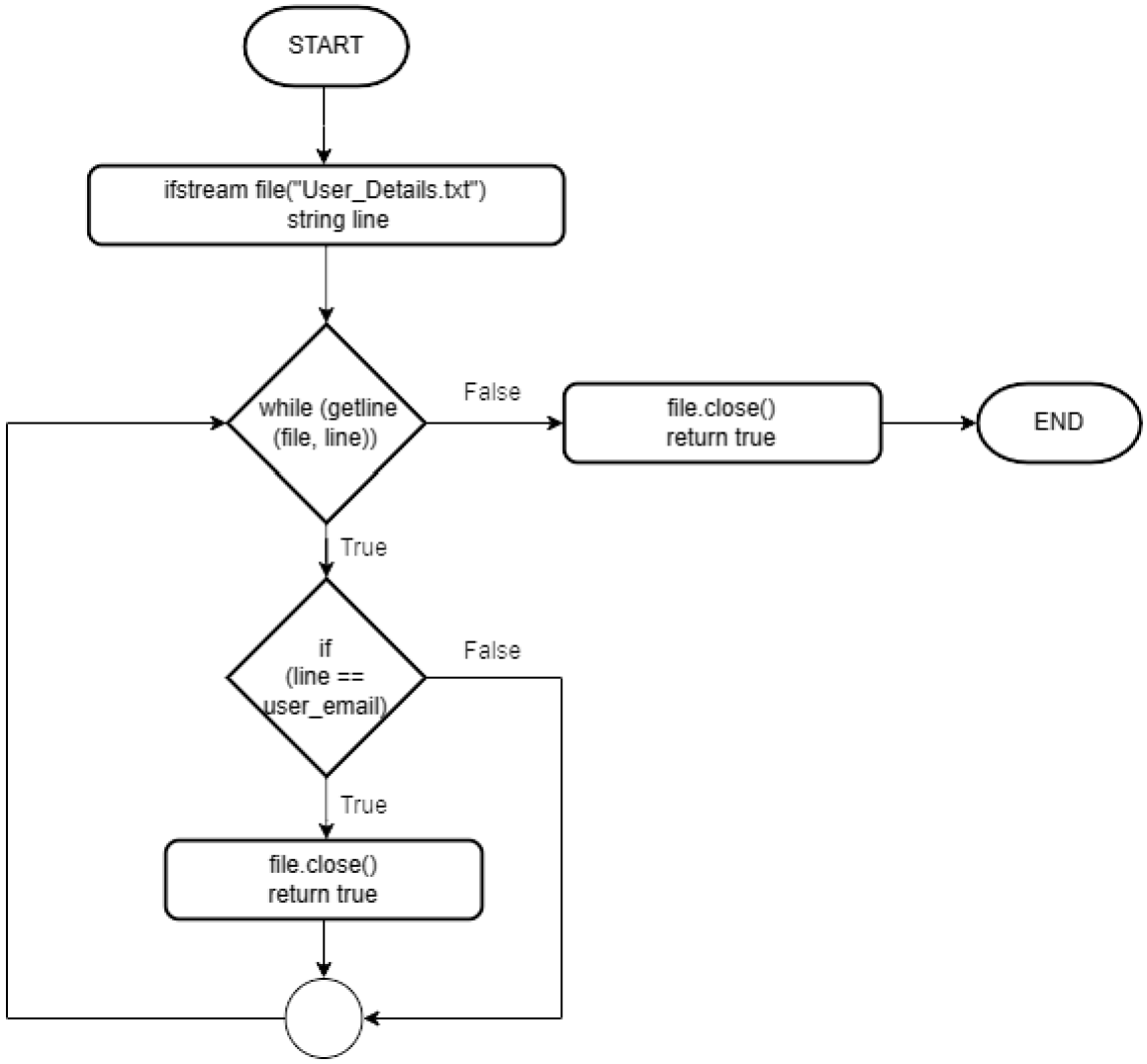
bool id_used(string user_id)



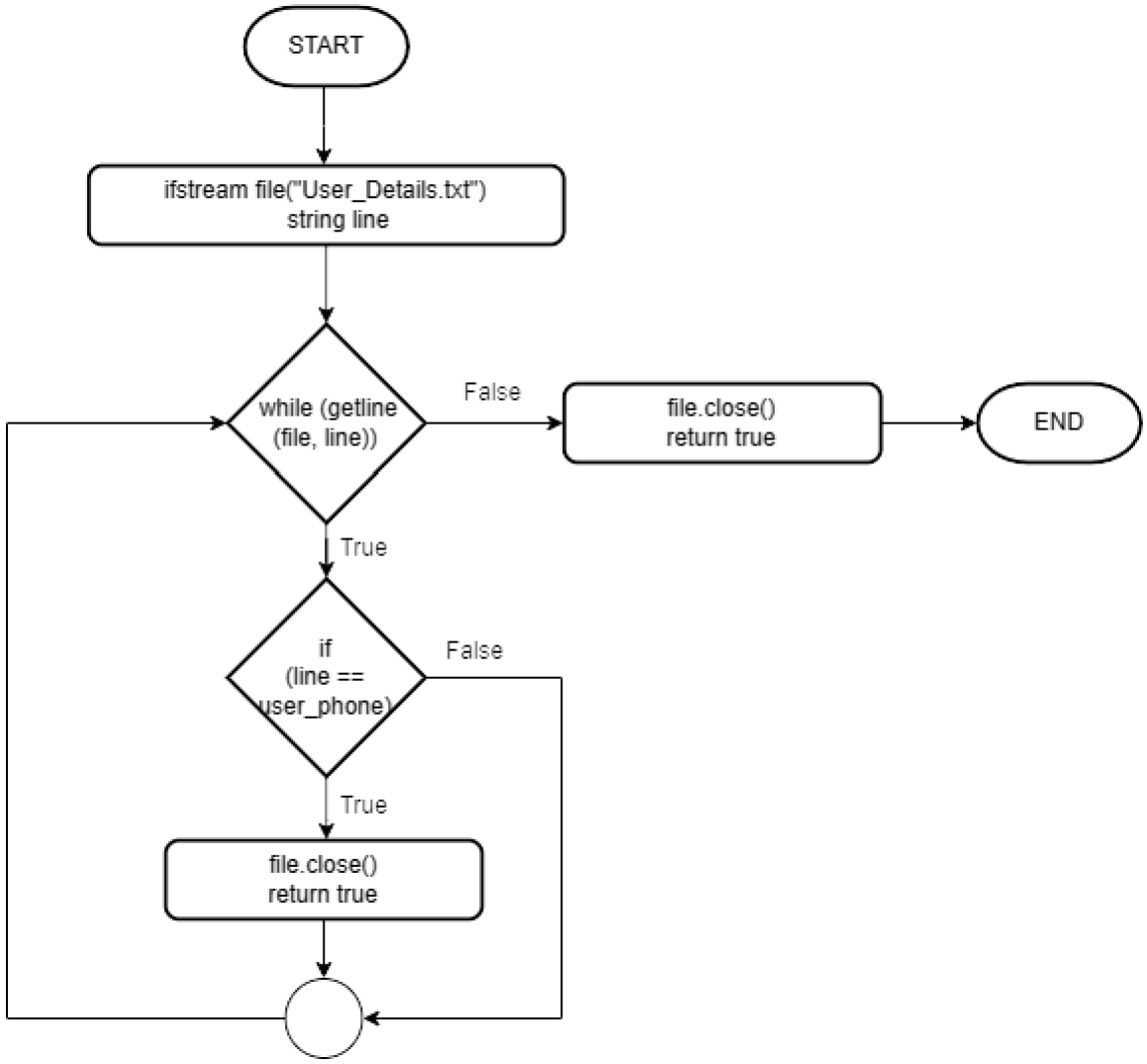
bool nric_used(string user_nric)



bool email_used(string user_email)



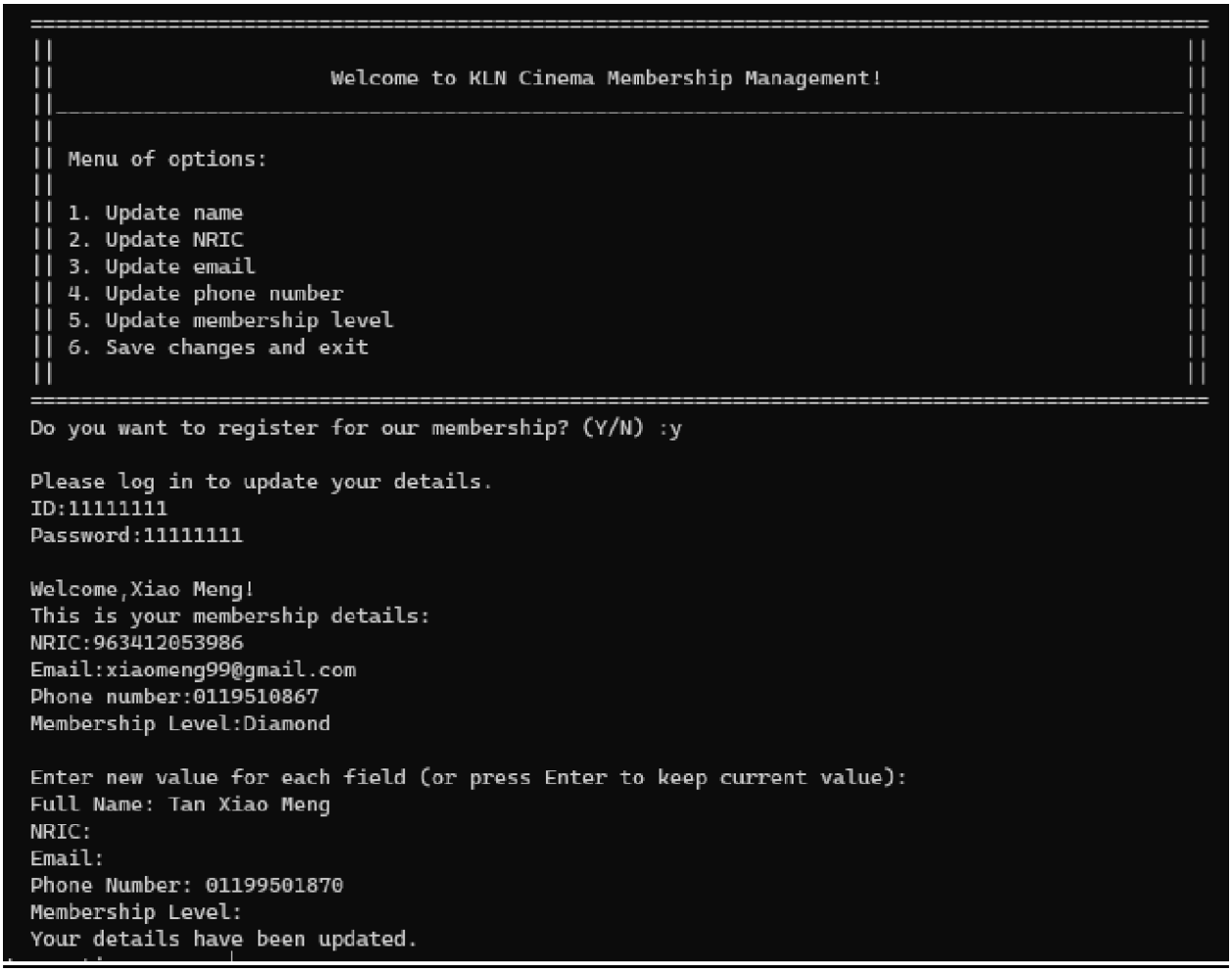
bool nric_used(string user_nric)



Test Case

1. **Users are allowed to register their membership.**

2. **Users are allowed to update their membership details.**



3. **Users are allowed to view synopsis of the movies.**

4. Users are allowed to book their tickets

```
Movie ID    Movie Name                          Time                 Director(s)                Language     Genre                          Ranking    Price(RM)    Time Available
A01         ANT-MAN AND THE WASP: QUANTUMANIA   2 Hours 05 Minutes   Peyton Reed                ENGLISH      Action / Adventure / Comedy    1          18.00        10:00AM-12:15PM
A02         DEMON SLAYER: KIMETSU NO YAIBO      1 Hour  50 Minutes   Haruo Sotozaki             JAPANESE     Animation                      2          16.50        12:30PM-14:30PM
A03         PULAU                               1 Hour  50 Minutes   Euho                       MALAY        Horror / Thriller              3          14.25        15:00PM-17:00PM
A04         65                                  1 Hour  35 Minutes   Scott Beck, Bryan Woods    ENGLISH      Science Fiction / Thriller     4          17.00        17:30PM-19:15PM
A05         DASARA                              2 Hours 40 Minutes   Srikanth Odhela            TAMIL        Action / Drama                 5          12.00        19:30PM-22:20PM

            ======================              ======================              ======================
            || 1 || Synopsis      ||            || 2 || Booking       ||            || 3 || Exit          ||
            ======================              ======================              ======================
                                          Option: 2
Please key in the movie ID that you want to book (<Q>uit to exit this process) : A01


_______                         ============================================================                    _______
| DOOR |                        ||                          SCREEN                          ||                   | DOOR |
|      |                        ============================================================                    |      |
                                ||              ANT-MAN AND THE WASP: QUANTUMANIA           ||
                                ============================================================

                                    01 02 03 04 05 06 07 08 09 10
                                  A #  #  #  #  #  #  #  #  #  #
                                  B #  #  #  #  #  #  #  #  #  #
                                  C #  #  #  #  #  #  #  #  #  #
                                  D #  #  #  #  #  #  #  #  #  #
                                  E #  #  #  #  #  #  #  #  #  #
                                  F #  #  #  #  #  #  #  #  #  #
                                  G #  #  #  #  #  #  #  #  #  #
                                  H #  #  #  #  #  #  #  #  #  #
                                  I #  #  #  #  #  #  #  #  #  #
                                  J #  #  #  #  #  #  #  #  #  #
Choose a seat that you prefer.
Choose a row : A
Choose the column (Please make sure is two digit): 08

The booking is recorded and you can check it in the cart

_______                         ============================================================                    _______
| DOOR |                        ||                          SCREEN                          ||                   | DOOR |
|      |                        ============================================================                    |      |
                                ||              ANT-MAN AND THE WASP: QUANTUMANIA           ||
                                ============================================================

                                    01 02 03 04 05 06 07 08 09 10
                                  A #  #  #  #  #  #  #  X  #  #
                                  B #  #  #  #  #  #  #  #  #  #
                                  C #  #  #  #  #  #  #  #  #  #
                                  D #  #  #  #  #  #  #  #  #  #
                                  E #  #  #  #  #  #  #  #  #  #
                                  F #  #  #  #  #  #  #  #  #  #
                                  G #  #  #  #  #  #  #  #  #  #
                                  H #  #  #  #  #  #  #  #  #  #
                                  I #  #  #  #  #  #  #  #  #  #
                                  J #  #  #  #  #  #  #  #  #  #
```

5. **Users are allowed to pay with different methods and enjoy a membership discount.**

```
Date: 26/4/2023                                                                                    Time: 14:25:25


Number of ticket(s) booking: 1
No discount is given. Please note that purchasing four tickets below does not qualify for additional discount.


Is the customer a member?  <Yes> / key in any key for no : yes


Customer ID : 11111111
Password  : 11111111
The customer is Diamond member. 15% discount is given.


=============================================================================================
Movie ID       Movie Name                                Seat    Price(RM)       Quantity
=============================================================================================
A01            ANT-MAN AND THE WASP: QUANTUMANIA         A08     18.00           x1
=============================================================================================
Subtotal Price    (RM)                                           18.00
Discount          (RM)                                          -0.00
Member Discount   (RM)                                          -2.70
=============================================================================================
Total Price       (RM)                                           15.30
Rounding Adjustment  (RM)                                        0.00
=============================================================================================
Total Price (after rounding adjustment)  (RM)                    15.30
=============================================================================================
What is the payment method of the customer? <E>-wallet / <C>ash / <C>redit <C>ard : C
Amount of the customer payed : 15.30
Is the payment done? <Y>es / key in any key to reset the payment method : Y
==================================
|| Find change (RM)||        0.00||
==================================
```

Source Code

```
include <iostream>
#include <fstream>
#include <string>
#include <cctype>
#include <cmath>
#include <iomanip>
#include <ctime>//for decoration purpose only
using namespace std;
//about option1-order
void system_interface();
void print1();
void option1();
void print2();
void print3();
void synopsis();
bool exit_page();

struct Movie
{
    string movie_ID;
    string movie_name;
    string time;
    string directors;
    string language;
    string genre;
    int ranking;
    double price;
    string slot;
};

const int max_movie = 5; //max exist 5 movie only
int array_movie(Movie movies[], const string&);
Movie movies[max_movie];
int num_movie = array_movie(movies, "Movie_information.txt");
void seat(int);
const int column = 10;
const int row = 10;
char seats[max_movie][row][column] = {};
void valid_of_seat(int, string);
void booking();
int num_of_book = 0;
string seat_code(string, char);
void save_into_cart(string, string, string);
void cart(string);

struct Cart
```

```
{
    string record_id;
    string record_name;
    string record_seat;
    double record_price;
};

const int max_line_in_cart = 500;
int array_cart(Cart carts[]);
Cart carts[max_line_in_cart];
bool edit_cart();
string delete_id_seat_code(string, string);
void delete_id_seat(string, string);
int line_in_cartfile = array_cart(carts);
// about option2- payment
void print4();
bool payment();
double discount();
double payment_method(double);
string method;
double customer_pay = 0;
double member_discount();
string membership;
//about option3-transaction record
void history_list();
bool history();
//about option4- membership
bool option4();
void membership_interface();
void acc_registration();
void acc_management();
void exit_program();
void clear_screen();

struct User
{
    string id;
    string password;
    string name;
    string nric;
    string email;
    string phone;
    string membership_level;
};

int main() {
```

```
    int option;
    bool enter = true;

    while (enter)
    {
        system_interface();
        cin >> option;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');// prevent invalid
variable type of input
        switch (option)
        {
        case 1:
        {
            option1();
            enter = true;
            break;
        }
        case 2:
        {
            enter = payment();
            break;
        }
        case 3:
        {
            enter = history();
            break;
        }
        case 4:
        {
            enter = option4();
            break;
        }
        case 5:
        {
            enter = exit_page();
            break;
        }
        default:
        {
            cout << "Invalid input. Please key in the number between \"1\" to \"6\".
" << endl;
            enter = true;
        }
        }
    }
    system("pause");
```

```
    return 0;
}

void system_interface()
{
    system("cls");// Clear the entire screen
    cout << "\n\n\n";
    cout << "\t\t
================================================================================
================================================================================"
<< endl;
    cout << "\t\t||
||" << endl;
    cout <<
"\t\t||
||" << endl;
    cout << "\t\t||                          .::::.                          //
____   _____       _____                   ____          _____ \t\t\t||" << endl;
    cout << "\t\t||                      .+%@@@@@@@*-                        //
|   | /    /       |    |                |    \\       |    |\t\t\t||" << endl;
    cout << "\t\t||:+#@@@@%*-          *@@+:     .-#@%:                     //
|   |/    /        |    |                |     \\      |    |\t\t\t||" << endl;
    cout << "\t\t||    *@@+:..:=#@%: #@#           -@@:                   //
|   '    /         |    |                |      \\     |    |\t\t\t||" << endl;
    cout << "\t\t||#@*  -@@-@@.         *@*          //
|       /          |    |                |       \\    |    |\t\t\t||" << endl;
    cout << "\t\t||  .@@.          %@+@@.            *@*               //
|      /           |    |                |        \\   |    |\t\t\t||" << endl;
    cout << "\t\t||@@=  :@@:#@#         -@@         //
|   ,   \\          |    |                |     __  \\__|    |\t\t\t||" << endl;
    cout << "\t\t||   .%@*:    .+@@=  *@%+:     .-#@%              //
|   |\\   \\         |    |_________       |    | \\          |\t\t\t||" << endl;
    cout << "\t\t||    :@@@@@@@@@@@%%%%@@@@@@@@@@@-      .=#%=    //
|   | \\   \\        |              |     |    |   \\        |\t\t\t||" << endl;
    cout << "\t\t||   *@@@@@@@@@@@@@@@@@@@@@@@@@@#  .=#@@@@@.  //
|__|  \\__\\         |______________|     |____|    \\_______|\t\t\t||" << endl;
    cout << "\t\t||  .@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*#@@@@@@@@
//
||" << endl;
    cout << "\t\t||  :@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.//
||" << endl;
    cout << "\t\t||  :@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.\\\\
\t\t\t||" << endl;
    cout << "\t\t||  :@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
\\\\                                          :+####=.  #  ==       *#: #######  =
=      +.\t\t\t||" << endl;
    cout << "\t\t||  :@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
\\\\                                        :#@%+=+*%* @@ #@#:     #@:
@@      :@+       .#%    =@%\t\t\t||" << endl;
```

```
    cout << "\t\t||  .@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*#@@@@@@@@@.
\\\\                                          .@@+         @@ #@@@%-  #@: @@       :@@#.
=@@%   -@@@#\t\t\t||" << endl;
    cout << "\t\t||   *@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#  .=#@@@@@.     \\\\
=@@.        @@ *@% #@% %@: @@@@@@# :@@@@=  *@@@%  :@@=#@*\t\t\t||" << endl;
    cout << "\t\t||    :*%%@@@@@@@@@@@@@@@@@@@@@%%*-       .=#%=
\\\\                                         .%@*.        @@ *@#  :#@@@: @@       :@%
@#@@  @%  .@@%  @@+\t\t\t||" << endl;
    cout << "\t\t||
\\\\                                         .*@@####@# @% *@#     :#@- @@       :@%
=@%: +@# .%@*=+++@@=\t\t\t||" << endl;
    cout << "\t\t||
\\\\                                          :+####=. += -+=       :: ####### .++
-+= -+=     :++\t\t\t||" << endl;
    cout << "\t\t||                                                      \\\\\\
||" << endl;
    cout << "\t\t||
\\\\
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t||
||" << endl;
    cout <<
"\t\t=============================================================================
=============================================================================
====" << endl;
    cout << "\n\n\n";
    cout << "\t\t\t\t\t ============================== \t
============================== \t\t ==============================" << endl;
    cout << "\t\t\t\t\t || 1 || Order                || \t || 2 || Payment
|| \t\t || 3 || Transaction Record  ||" << endl;
    cout << "\t\t\t\t\t ============================== \t
============================== \t\t ==============================" << endl;
    cout << "\n" << endl;
    cout << "\t\t\t\t\t\t\t ============================== \t
==============================" << endl;
    cout << "\t\t\t\t\t\t\t || 4 || Membership           || \t || 5 || Exit
||" << endl;
    cout << "\t\t\t\t\t\t\t ============================== \t
==============================" << endl;
    cout << "\n\t\t\t\t\t\t\t\t\t\t\t\tOption : ";
}

void print1()
{
    system("cls");
    cout << " ==========================================" << endl;
    cout << " || 1 ||  Book Ticket(s)                   ||" << endl;
```

```
    cout << " ==========================================" << endl;
    cout << " || 2 ||  Cart                              ||" << endl;
    cout << " ==========================================" << endl;
    cout << " || 3 ||  Exit                              ||" << endl;
    cout << " ==========================================" << endl;
}

void option1()
{
    bool option1 = true;
    int choice, choice1;
    int count = 0;
    char yes_no;
    string line, booking1;

    while (option1)
    {
        print1();// call function

        cout << "Please choose the option that you want to conduct : ";
        cin >> choice;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');// prevent invalid
variable type of input

        switch (choice)
        {
        case 1:
        {
            bool a = true;
            while (a)
            {
                print2();// call function
                array_movie(movies, "Movie_information.txt");// call function

                cout << left;
                cout << setw(12) << "Movie ID" << setw(41) << "Movie Name" <<
setw(26) << "Time" << setw(31) << "Director(s)" << setw(15) << "Language" << setw(33)
<< "Genre"
                    << setw(13) << "Ranking" << setw(12) << "Price(RM)" << "Time
Available" << endl;

                for (int i = 0; i < 5; i++)
                {
                    cout << setprecision(2) << fixed;
                    cout << setw(12) << movies[i].movie_ID << setw(41) <<
movies[i].movie_name << setw(26) << movies[i].time << setw(31) << movies[i].directors
<< setw(15)
```

```
                        << movies[i].language << setw(33) << movies[i].genre <<
setw(13) << movies[i].ranking << setw(12) << movies[i].price << movies[i].slot <<
endl;
                }
                cout << endl << endl;

                print3();// call function

                cin >> choice1;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');

                switch (choice1)
                {
                case 1:
                {
                    cout << "\nDo you want to see the synopsis of the movie? (<Y>es /
Key in any key to exit) ";
                    cin >> yes_no;

                    if (toupper(yes_no) == 'Y')
                    {
                        synopsis();//call function
                    }
                    else
                    {
                        cin.clear();
                        cin.ignore(numeric_limits<streamsize>::max(), '\n');
                        a = true;
                    }
                    break;
                }
                case 2:
                {
                    booking();//call function
                    a = true;
                    option1 = true;
                    break;
                }
                case 3:
                {
                    // call function
                    a = exit_page();
                    break;
                }
                default:
                {
```

```
                cout << "Invalid input!!! Please key in again." << endl;
                a = true;
            }
            }
        }
        break;
    }

    case 2:
    {
        system("cls");
        bool b = true;

        while (b)
        {
            cart("cart.txt");// call function
            b = edit_cart();// call function
        }
        break;
    }

    case 3:
    {
        option1 = exit_page();
        break;
    }

    default:
    {
        cout << "Invalid Input" << endl;
    }
    }
}
}

bool exit_page()
{
    char exit;

    cout << "Do you want to exit this page? (<Y>es or key in any key for No) ";
    cin >> exit;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    if (toupper(exit) == 'Y')
    {
        return false;
```

```
    }
    else
    {
        return true;
    }
}

void print2()
{
    system("cls");
    cout <<
"================================================================================
========================" << endl;
    cout << "||    _________________    _________________    _________________
_________________       ______      ||" << endl;
    cout << "||   |                 |    |                 |   |                 |
|                |     |      |     ||" << endl;
    cout << "||   |______     _____|    |    ______     |   |     ______     |
|   ____________|     |      |     ||" << endl;
    cout << "||         |    |         |   |     |    |   |    |     |    |
|   |                 |      |     ||" << endl;
    cout << "||         |    |         |   |     |    |   |    |_____|    |
|   |____________     |      |     ||" << endl;
    cout << "||         |    |         |   |     |    |   |     ___________|
|                |     |      |     ||" << endl;
    cout << "||         |    |         |   |     |    |   |    |
|___________     |     |      |     ||" << endl;
    cout << "||         |    |         |   |     |    |   |    |
|   |     |______|     ||" << endl;
    cout << "||         |    |         |   |_____|    |   |    |
___________|   |                    ||" << endl;
    cout << "||         |    |         |                 |   |    |
|                |       ______      ||" << endl;
    cout << "||         |____|         |_________________|   |____|
|________________|      |_____|     ||" << endl;
    cout << "||
||" << endl;
    cout <<
"================================================================================
========================" << endl;
    cout << "\n\n\n" << endl;
}

void print3()
{
    cout <<
"\t\t========================\t\t========================\t\t=====================
===" << endl;
```

```
    cout << "\t\t|| 1 || Synopsis       ||\t\t|| 2 || Booking        ||\t\t|| 3 ||
Exit           ||" << endl;
    cout <<
"\t\t=======================\t\t=======================\t\t=====================
===" << endl;
    cout << "\t\t\t\t\t\t\tOption: ";
}

int array_movie(Movie movies[], const string& filename)
{
    int num_movie = 0;
    ifstream movie_file(filename);

    if (movie_file.is_open())
    {
        string line;
        // store the movie information from file and store it in struc in array
        while (getline(movie_file, line) && num_movie < max_movie)
        {
            Movie movie;
            movie.movie_ID = line.substr(0, 11);
            movie.movie_name = line.substr(11, 40);
            movie.time = line.substr(52, 25);
            movie.directors = line.substr(78, 30);
            movie.language = line.substr(109, 15);
            movie.genre = line.substr(125, 33);
            movie.ranking = stoi(line.substr(158, 1));
            movie.price = stod(line.substr(163, 5));
            movie.slot = line.substr(172, 15);
            movies[num_movie] = movie;
            num_movie++;
        }
    }

    else
        cout << "Not recorded found." << endl;

    movie_file.close();

    return num_movie;// in order to know how many movie in file
}

void synopsis() //view synopsis module
{
    string line, synopsis_line, movie_ID;
    int count = 0;
    char continues;
```

```
    bool a = true;

    while (a)
    {
        cout << "\nKey in the ID of the movie that you want to see the synopsis : ";
        cin >> movie_ID;

        if (!isalpha(movie_ID[0]) || !isdigit(movie_ID[1]) || !isdigit(movie_ID[2]))
//check the format of the movie id first must be alphabet while second and third must
be the digit
        {
            cout << "\nInvalid input for movie ID!!! Please key in again.\n" << endl;
            a = true;
        }
        else
        {
            ifstream moviefile("Movie_information.txt");

            if (moviefile.is_open())
            {
                bool found_movie_id = false;

                cout << endl << endl << endl;
                moviefile.seekg(0, ios::beg);  // reset file pointer to beginning

                for (int i = 0; i < num_movie; i++)
                {
                    getline(moviefile, line);
                    // find the movie id have stored in the struc in array or not, if
yes will not return the value of npos and also found have the
                    //  existence of "." or not, it is necessary no, and then just
can combine become the name of a txt file that consists synopsis
                    if (movies[i].movie_ID.find(movie_ID, 0) != string::npos &&
movie_ID.find('.') == string::npos)
                    {
                        string filename = movie_ID + ".txt";
                        ifstream synopsis_file(filename);

                        while (getline(synopsis_file, synopsis_line))
                            cout << synopsis_line << endl;// read the file contains
synopsis

                        found_movie_id = true;  // set variable to true in order to
check validation of movie_id
                        break;  // exit the loop after finding the movie ID once
since it is for loop
                    }
                }
```

```
                if (!found_movie_id)
                    cout << "Invalid movie ID!! Please check the movie ID and key in
again. " << endl;

                moviefile.close();

                cout << "\nDo you continue to see another synopsis of the movie?
(<Q>uit to exit / Key in any key to continue) : ";
                cin >> continues;
                cout << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                if (toupper(continues) == 'Q')
                    a = false;
                else
                    a = true;
            }
        }
    }
}

void booking() // booking module
{
    bool a = true, find = false;
    string movie_id, line;
    string seat_code;
    char bookcode[3] = {};

    for (int i = 0; i < num_movie; i++)
    {
        for (int j = 0; j < row; j++)
        {
            for (int k = 0; k < column; k++)
            {
                seats[i][j][k] = '#';
            }
        }
    }

    ifstream infile("booking.txt");
    if (infile.is_open())
    {
        while (infile >> seat_code)
        {
            int movieIndex = seat_code[0] - '0';//deduct ASCII value of '0' to take
integer , this is the movie type
```

```
            char row = seat_code[1];//position of row
            int col = stoi(string(1, seat_code[2])) * 10 + stoi(string(1,
seat_code[3])) - 1;//multiple 10 to get tens-digit and plus ones-digit to get the
last two digit that is the position of column and then minus 1 to get the position of
array
            if (movieIndex >= 0 && movieIndex <= num_movie)
            { // Check if movieIndex is within bounds
                seats[movieIndex - 1][toupper(row) - 'A'][col] = 'X'; // X means seat
is taken
            }
        }
        infile.close();
    }

    while (a)
    {
        cout << "Please key in the movie ID that you want to book (<Q>uit to exit
this process) : ";
        cin >> movie_id;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        if (toupper(movie_id[0]) == 'Q')
        {
            a = false;
        }
        else if (!isalpha(movie_id[0]) || !isdigit(movie_id[1])
|| !isdigit(movie_id[2]))
        {
            cout << "Invalid input for movie ID!!! Please key in again." << endl;
            a = true;
        }
        else
        {

            cout << endl << endl;
            for (int i = 0; i < num_movie; i++)
            {
                if (movies[i].movie_ID.find(movie_id, 0) != string::npos)
                {

                    for (int j = 0; j < 3; j++)
                    {
                        bookcode[j] = movie_id[j];
                    }
                    int number_of_id = stoi(string(1, bookcode[1])) * 10 +
stoi(string(1, bookcode[2]));//change the string into integer, to get the last two
number of movie id, multiple 10 in order to get tens-digit and then plus one-digit
```

```
                    seat(number_of_id); //call function
                    valid_of_seat(number_of_id, movie_id);// call function

                    find = true;
                    char continues;
                    bool c = true;

                    while (c)
                    {
                        cout << "\nDo you want to book another ticket? <Y>es / <N>o :
";
                        cin >> continues;
                        cin.clear();
                        cin.ignore(numeric_limits<streamsize>::max(), '\n');
                        cout << endl << endl;
                        if (toupper(continues) == 'Y')
                        {
                            c = false;
                            a = true;
                        }
                        else if (toupper(continues) == 'N')
                        {
                            c = false;
                            a = false;
                        }
                        else
                        {
                            cout << "Please key in 'Y' or 'N' to proceed the
system.\n" << endl;
                            c = true;
                        }
                    }

                }
            }
            if (!find)
            {
                cout << "Invalid movie ID!!! Please key in again.\n" << endl;
            }
        }
    }
}
void seat(int num_of_id)// display seat
{
```

```
    cout <<
"________\t\t\t\t=========================================================================\
t\t\t\t________" << endl;
    cout << "| DOOR |\t\t\t\t||                                SCREEN
||\t\t\t\t| DOOR |" << endl;
    cout << "|
|\t\t\t\t=========================================================================\t\t\t\t|
|" << endl;
    cout << "        \t\t\t\t||                " << movies[num_of_id - 1].movie_name
<< "       ||\t\t\t\t        " << endl;
    cout << "
\t\t\t\t=========================================================================\t\t\t\t
" << endl << endl;
    cout << "        \t\t\t\t\t\t 01 02 03 04 05 06 07 08 09 10" << endl;

    for (int i = 0; i < row; i++)
    {
        cout << "\t\t\t\t\t\t\t" << char('A' + i) << " ";

        for (int j = 0; j < column; j++)
        {
            cout << seats[num_of_id - 1][i][j] << "  ";
        }
        cout << endl;

    }
}

void valid_of_seat(int number_of_id, string movie_id)
{
    string movie_id1 = movie_id;
    char choose_row;
    string choose_column;// type of variable is char because after need to use
isdigit()

    cout << "Choose a seat that you prefer." << endl;
    cout << "Choose a row : ";
    cin >> choose_row;
    cout << "Choose the column (Please make sure is two digit): ";
    cin >> choose_column;
    choose_row = toupper(choose_row);

    if (!choose_row)
    {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Invalid Input. Please key in again." << endl;
    }
```

```
    if (isalpha(choose_row) && isdigit(choose_column[0]) && isdigit(choose_column[1])
&& choose_column.length() == 2 && choose_row <= 'J' && choose_column[0] <= '1')
    {
        ofstream outfile("booking.txt", ios::app);

        char choose_column_char[2] = {};
        for (int i = 0; i < 2; i++)
        {
            choose_column_char[i] = choose_column[i];
        }
        int choose_column_int = stoi(string(1, choose_column_char[0])) * 10 +
stoi(string(1, choose_column_char[1]));// calculate value of column in int type

        if (choose_column_int <= 10)
        {
            if (seats[number_of_id - 1][choose_row - 'A'][choose_column_int - 1] ==
'X')
            {
                cout << "\nThat seat is already booked by other people, Please choose
a different seat.\n" << endl;
            }
            else if (seats[number_of_id - 1][choose_row - 'A'][choose_column_int - 1]
== '#') //book the seat
            {
                seats[number_of_id - 1][choose_row - 'A'][choose_column_int - 1] =
'X';
                cout << "\nThe booking is recorded and you can check it in the
cart\n" << endl;

                outfile << number_of_id << choose_row << choose_column <<
endl;//store seat code into file
                outfile.close();

                string column_row = choose_column.insert(0, 1, choose_row); // get
the seat position

                save_into_cart(movie_id1, "cart.txt", column_row); //call function
                seat(number_of_id); //function that use to display the seat
            }
        }
        else
            cout << "\nPlease key in the number within 01 - 10.\n" << endl;
    }
    else
        cout << "\nPlease check the input. Make sure the row is alphabet within A to
J whereas column is digit within 01 to 10.\n" << endl;
}
```

```
int array_cart(Cart carts[])
{
    int line_in_cartfile = 0;

    ifstream cartfile("cart.txt");
    if (cartfile.is_open())
    {
        string line;
        while (getline(cartfile, line) && line_in_cartfile < max_line_in_cart)
        {
            Cart cart;
            cart.record_id = line.substr(0, 15);
            cart.record_name = line.substr(15, 56);
            cart.record_seat = line.substr(56, 3);
            cart.record_price = stod(line.substr(64, 5));
            carts[line_in_cartfile] = cart;
            line_in_cartfile++;
        }
    }
    cartfile.close();
    return line_in_cartfile;
}

void save_into_cart(string movie_id1, string filename, string column_row)//save the
booking into cart
{
    array_movie(movies, "Movie_information.txt");
    string name;
    double price_movie;
    int line_in_cart = array_cart(carts);

    for (int i = 0; i < num_movie; i++)
    {
        if (movies[i].movie_ID.find(movie_id1) != string::npos)//if the movie id
found in the movie information file
        {
            name = movies[i].movie_name;
            price_movie = movies[i].price;
            for (int j = 0; j <= line_in_cart; j++)
            {
                if (carts[j].record_seat.find(column_row) == string::npos ||
carts[j].record_id.find(movie_id1) == string::npos) // if the movie id is not found
or the position of the seat is not found accordingly in the cart file it will save
the booking into the cart.
                {
                    ofstream writefile(filename, ios::app);
                    if (writefile.is_open())
```

```
                    {
                        writefile << setprecision(2) << fixed << left
                            << setw(15) << movie_id1 << setw(41) << name <<
column_row << right << setw(10) << price_movie << endl;
                    }
                    writefile.close();
                    break;
                }
            }
        }
    }
}

void cart(string filename)//review the cart
{
    string line;
    cout << setprecision(2) << fixed << left;
    cout << setw(15) << "Movie ID" << setw(41) << "Movie Name" << setw(8) << "Seat"
<< setw(9) << "Price(RM)" << "\tQuantity" << endl;
    cout <<
"================================================================================
===========" << endl;

    ifstream printfile(filename);
    if (printfile.is_open())
    {
        // check the stream has reached the end of the file or not if yes
printfile.peek() will return EOF means that there are no word exist in the file
        if (printfile.peek() == EOF)
        {
            cout << "\t\t==============================" << endl;
            cout << "\t\t||    Not Recorded Found    ||" << endl;
            cout << "\t\t==============================" << endl;
        }
        else
        {
            while (getline(printfile, line))
            {
                cout << line << "\t\tx1" << endl;
            }
        }
        printfile.close();
    }
    else
    {
        cout << "File is not found" << endl;
    }
}
```

```
bool edit_cart() //delete booking module
{

    cout << endl << endl << endl;
    cout << "========================\t\t========================" << endl;
    cout << "|| 1 || Delete        ||\t\t|| 2 || Exit        ||" << endl;
    cout << "========================\t\t========================" << endl;
    cout << "                        \tOption: ";

    int cart_option;
    bool b;

    cin >> cart_option;
    switch (cart_option)
    {
    case 1:
    {
        string delete_id, delete_seat;
        cout << endl << endl << endl;
        cout << "Please Key in the movie id and accordingly seat that you want to
delete. " << endl;
        cout << "Movie ID : ";
        cin >> delete_id;
        cout << "Seat: ";
        cin >> delete_seat;
        cout << endl << endl;

        if (!cin)
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }

        delete_id_seat(delete_id, delete_seat);//call function
        return true;
        break;
    }
    case 2:
    {
        b = exit_page();//call function
        return b;
        break;
    }
    default:
    {
        if (!cart_option)
```

```
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\nInvalid Input. Please key in again.\n\n" << endl;
        }
        else if (cart_option != 1 && cart_option != 2)
        {
            cout << "\nInvalid Input. Please key in \"1\" / \"2\" to conduct the
process.\n\n" << endl;
        }
        return true;
    }
    }
}
void delete_id_seat(string delete_id, string delete_seat)
{
    int num_book_in_cart = array_cart(carts);
    string cart_array[500] = {}, ticket_book[500] = {};//max ticket is 500(5x10x10)
    bool valid_info = false;
    string delete_seat_code;

    ifstream infile("cart.txt");
    for (int i = 0; i < num_book_in_cart; i++) //store the contents of cart file in
array
    {
        string line;
        getline(infile, line);
        cart_array[i] = line;
    }

    ifstream infile2("booking.txt"); //in order to store contents of booking file in
array

    int count_line = 0;
    string line1;
    while (getline(infile2, line1))
        count_line++;

    infile2.clear(); // reset the file stream to the beginning
    infile2.seekg(0, ios::beg);

    for (int j = 0; j < count_line; j++)
    {
        getline(infile2, ticket_book[j]);
    }
```

```
    if (isalpha(delete_id[0]) && isdigit(delete_id[1]) && isdigit(delete_id[2]) &&
isalpha(delete_seat[0]) && isdigit(delete_seat[1]) && isdigit(delete_id[2]) &&
delete_id.length() == 3 && delete_seat.length() == 3)
    {
        ofstream outfile("cart.txt");
        ofstream outfile2("booking.txt");

        for (int k = 0; k < num_book_in_cart; k++)
        {
            if (infile.is_open() && carts[k].record_id.find(delete_id, 0) ==
string::npos || carts[k].record_seat.find(delete_seat, 0) == string::npos)//if delete
id or delete seat not found in the cart file
            {
                delete_seat_code = delete_id_seat_code(delete_seat, delete_id);//
call the function to get the complete seat code that want to delete
                outfile << cart_array[k] << endl;
            }
        }

        for (int l = 0; l < count_line; l++)
        {
            if (infile2.is_open() && ticket_book[l].find(delete_seat_code, 0) ==
string::npos)//if the seat code want to delete is not found in bookig file
            {
                outfile2 << ticket_book[l] << endl;
                valid_info = true;
            }
        }

        outfile.close();
        outfile2.close();
    }

    if (!valid_info)
    {
        cout << "\nThe movie id or seat that you want to delete is not found in the
cart. Please make sure you have done the booking or not.\n\n" << endl;
    }

    infile.close();
    infile2.close();
}

string delete_id_seat_code(string delete_seat, string delete_id)
{
    char delete_seatcode[3] = {};

    for (int j = 0; j < 3; j++)
```

```
    {
        delete_seatcode[j] = delete_id[j];
    }
    int number_of_delete_id = stoi(string(1, delete_seatcode[1])) * 10 +
stoi(string(1, delete_seatcode[2]));
    char number = number_of_delete_id + '0';//get the movie type
    string delete_seat_code = delete_seat.insert(0, 1, number);
    return delete_seat_code;
}

void print4()
{
    system("cls");
    cout <<
"================================================================================
==================================\n";
    cout << "||
||\n";
    cout << "||
||\n";
    cout << "||                    _______          _     __    __   ___    ___    ______
__       _    ________                    ||\n";
    cout << "||                  |  ____\\     / \\    \\ \\  / /  |   \\  /   |  |
____|  |   \\   | |  |___  __|                      ||\n";
    cout << "||                  | |___| |   / _ \\    \\ \\/ /   | |\\ \\/ /| |  |
|____   | |\\ \\  | |     | |                        ||\n";
    cout << "||                  |  ____/   / /_\\ \\    |  |    | | \\  / | |  |
___|   | | \\ \\ | |     | |                         ||\n";
    cout << "||                  | |       / ______\\   |  |    | |  \\/  | |  |
|_____  | |  \\ \\| |     | |                         ||\n";
    cout << "||                  |_|      /_/     \\_\\  |__|    |_|      |_|
|______|  |_|   \\___|     |_|                        ||\n";
    cout << "||
||\n";
    cout << "||
||\n";
    cout << "||
||\n";
    cout <<
"================================================================================
==================================\n";


    cout <<
"================================================================================
==================================" << endl;
    cout << "||
||" << endl;
```

```
    cout << "||                                                  ~  Discount Menu
~                                          ||" << endl;
    cout <<
"||_____________________________________________________________________________________________
_____________________________||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "||        __________    __     __        |     __       _____        __
___  |    __     __________    __     __ ||" << endl;
    cout << "||       |  _______|   |__|   / /        |    |  |    / _____\\      |__|
/ /   |   |  |   |  _______|   |__|   / /  ||" << endl;
    cout << "||       | |                / /         |    |  |   / /    \\ \\
/ /    |   |  |   | |                 / /   ||" << endl;
    cout << "||       | |_____          / /          |    |  |  | |      | |
/ /     |   |  |   | |_____          / /    ||" << endl;
    cout << "||       |_______ \\        / /           |    |  |  | |      | |
/ /      |   |  |   |_______ \\        / /     ||" << endl;
    cout << "||              \\  \\      / /            |    |  |  | |      | |
/ /       |   |  |          \\  \\      / /      ||" << endl;
    cout << "||               | |     / /             |    |  |  | |      | |      / /
|   |  |          | |     / /       ||" << endl;
    cout << "||        ______/  /    / /   __         |    |  |   \\ \\____/ /      /
/   __    |   |  |    ______/  /    / /   ___  ||" << endl;
    cout << "||       |________/    /_/   |__|        |    |__|    \\_____/      /_/
|__|   |   |__|   |________/    /_/   |__|  ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout <<
"||_____________________________________|_____________________________________|____
_____________________________||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "|| ~ For buying the 4 to 6 tickets      |  ~ For buying the 7 tickets or
above  |  ~ For Diamond Member Only          ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
    cout << "|| ~ For Gold Member                     |  ~ For Platinum Member
|                                  ||" << endl;
    cout << "||                                        |
|                                  ||" << endl;
```

```
    cout << "||                                          |
|                                    ||" << endl;
    cout <<
"=================================================================================
==================================" << endl;
    cout << endl << endl << endl;
}

bool payment()//payment module
{
    print4();//call function
    bool a = true;
    int line_in_cartfile = array_cart(carts);
    time_t now = time(nullptr);
    tm localTime;
    localtime_s(&localTime, &now);
    // display current date and time for decoration
    cout << left << "Date: " << localTime.tm_mday << "/" << localTime.tm_mon + 1 <<
"/" << localTime.tm_year + 1900;
    cout << right << setw(96) << "Time: " << setfill('0') << setw(2) <<
localTime.tm_hour << ":" << setfill('0') << setw(2) << localTime.tm_min << ":" <<
setfill('0') << setw(2) << localTime.tm_sec << endl;
    cout << setfill(' ');//do not want to apply setwill after display time
    cout << endl << endl;

    ifstream file("cart.txt");
    if (file.is_open())
    {
        if (file.peek() == EOF)
        {
            cout << "\t\t==============================" << endl;
            cout << "\t\t||    Not Recorded Found    ||" << endl;
            cout << "\t\t==============================" << endl;
            a = false;
        }
        else
            a = true;
    }
    else
    {
        cout << "File is not open." << endl;
        a = false;
    }
    file.close();

    while (a)
    {
        double discount_rate = discount();
```

```
        cout << endl << endl;
        double member_discount_rate = member_discount();

        cout << endl << endl;
        cout <<
"================================================================================
===========" << endl;
        cart("cart.txt");
        cout <<
"================================================================================
===========" << endl;

        double subtotal_price = 0;

        for (int i = 0; i < line_in_cartfile; i++)
        {
            subtotal_price += carts[i].record_price;
        }

        double discount_price = subtotal_price * discount_rate;
        double member_discount_price = subtotal_price * member_discount_rate;

        double total_price = subtotal_price - round((discount_price * 100)) / 100 -
round((member_discount_price * 100)) / 100;// multiple 100 and divide 100 and then
round them to get 2 decimal places
        // rounding adjustment
        int sen = round(fmod(total_price * 100, 10));
        if (sen % 10 >= 5)
            sen += 10;

        sen = sen - (sen % 10);

        double total_price_after_round = total_price - (fmod(total_price * 100, 10) /
100) + (sen / 100.0);

        cout << fixed << showpoint << setprecision(2) << left;
        cout << setw(64) << "Subtotal Price    (RM)" << subtotal_price << endl;
        cout << setw(63) << "Discount         (RM)" << setw(1) << '-' << setw(5) <<
discount_price << endl;
        cout << setw(63) << "Member Discount  (RM)" << setw(1) << '-' << setw(5) <<
member_discount_price << endl;
        cout <<
"================================================================================
===========" << endl;
        cout << setw(64) << "Total Price      (RM)" << total_price << endl;
        cout << setw(64) << "Rounding Adjustment  (RM)" << total_price_after_round -
total_price << endl;
```

```
        cout <<
"================================================================================
==========" << endl;
        cout << setw(64) << "Total Price (after rounding adjustment)  (RM)" <<
total_price_after_round << endl;
        cout <<
"================================================================================
==========" << endl;

        double find_change = 0;
        find_change = payment_method(total_price_after_round);//call function and
return value of change that need to find for customer

        string transaction_id;
        int number_of_file = 0;
        string filename;

        for (int i = 0; i <= 500; i++)//  max number of ticket can book is 500
        {
            if (i >= 0 && i <= 9)
            {
                transaction_id = "000" + to_string(i);
            }
            else if (i >= 10 && i <= 99)
            {
                transaction_id = "00" + to_string(i);
            }
            else if (i >= 100)
            {
                transaction_id = "0" + to_string(i);
            }
            filename = "transacrtion_id_" + transaction_id + ".txt";

            ifstream count_file(filename);

            if (count_file.is_open())
                number_of_file++;
            else if (count_file.fail())//save receipt to the txt file
            {
                ofstream receipt_in_file(filename);
                receipt_in_file <<
"================================================================================
==========" << endl;
                receipt_in_file << "|| Date: " << localTime.tm_mday << "/" <<
localTime.tm_mon + 1 << "/" << localTime.tm_year + 1900;
                receipt_in_file << setw(65) << "Time: " << setfill('0') << setw(2) <<
localTime.tm_hour << ":" << setfill('0') << setw(2) << localTime.tm_min << ":" <<
setfill('0') << setw(2) << localTime.tm_sec << "  ||" << endl;
```

```
                receipt_in_file << setfill(' ');
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << "||                                Transaction ID :
" << left << setw(44) << transaction_id << "||" << endl;
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << fixed << showpoint << setprecision(2) << left;
                receipt_in_file << "|| " << setw(15) << "Movie ID" << setw(41) <<
"Movie Name" << setw(8) << "Seat" << setw(9) << "Price(RM)" << "\tQuantity" << "
||" << endl;
                ifstream cart("cart.txt");
                string line;
                while (getline(cart, line))
                {
                    receipt_in_file << "|| " << line << "\tx1" << "\t     ||" <<
endl;
                }
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << setw(68) << "Subtotal Price   (RM)" <<
subtotal_price << endl;
                receipt_in_file << setw(67) << "Discount         (RM)" << setw(1) <<
'-' << setw(5) << discount_price << endl;
                receipt_in_file << setw(67) << "Member Discount  (RM)" << setw(1) <<
'-' << setw(5) << member_discount_price << endl;
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << setw(68) << "Total Price      (RM)" << total_price
<< endl;
                 receipt_in_file << setw(68) << "Rounding Adjustment  (RM)" <<
total_price_after_round - total_price << endl;
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << setw(68) << "Total Price (after rounding
adjustment)  (RM)" << total_price_after_round << endl;
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << setw(68) << "Payment method" << method << endl;
                if (method == "E-wallet" || method == "Credit Card")
                    customer_pay = total_price_after_round;
                receipt_in_file << setw(68) << left << "Customer Pay    (RM)" <<
customer_pay << endl;
```

```
                receipt_in_file << setw(68) << left << "Find Change     (RM)" <<
find_change << endl;
                receipt_in_file <<
"================================================================================
===========" << endl;
                receipt_in_file << "                            ThanK you. Please come
again. ^_^\n\n\n                                  " << endl;

                cart.close();
                receipt_in_file.close();
                count_file.close();

                ofstream history_file;//store several pieces of information that
required to txt file, just give the user can see the brief of record in the
transaction record part
                history_file.open("history.txt", ios::app);
                if (history_file.is_open())
                {
                    history_file << fixed << showpoint << setprecision(2) << left;
                    history_file << transaction_id << "\t\t" << customer_pay <<
"\t\t" << method << endl;
                }
                history_file.close();
                break;
            }
        }

        ofstream delete_content_of_file("cart.txt");//Delete the contents of the cart
file because different people have their own cart when booking, so the cart file
needs to delete and after the payment, the transaction would be recorded.
        delete_content_of_file.close();

        bool  b  =  true;
        char print_receipt;
        while (b)

        {
            cout << "\n\n\nDo you want to print the receipt? <Y>es / key in any key
to Quit : ";
            cin >> print_receipt;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            if (toupper(print_receipt) == 'Y')
            {
                ifstream print(filename);//print receipt from txt file
                string line;
                while (getline(print, line))
                    cout << line << endl;
```

```
                b = true;
                print.close();
            }
            else
            {
                b = false;
                return true;
            }
        }
    }
    char any;
    cout << "\nPlease go to book the ticket(s) first. Thank you ^_^.\n" << endl;
    cout << "Key in any key to Quit : ";
    cin >> any;// in order to quit the page only if no booking in the cart
    return true;

}

double discount()// discount module
{
    int line_in_cartfile = array_cart(carts);
    cout << "Number of ticket(s) booking: " << line_in_cartfile << endl;
    if (line_in_cartfile >= 4 && line_in_cartfile < 7)
    {
        cout << "Discount 5% for total price is applied." << endl;
        return 0.05;
    }
    else if (line_in_cartfile >= 7)
    {
        cout << "Discount 10% for total price is applied." << endl;
        return 0.1;
    }
    else
    {
        cout << "No discount is given. Please note that purchasing four tickets below
does not qualify for additional discount. " << endl;
        return 0.0;
    }
}

double member_discount()// member discount
{
    char yes_no;
    string customer_id, password, membership;
    bool a = true, valid_member = true;

    while (a)
```

```
    {
        cout << "Is the customer a member?  <Yes> / key in any key for no : ";
        cin >> yes_no;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        if (toupper(yes_no) == 'Y')
        {
            cout << endl << endl;
            cout << "Customer ID : ";
            cin >> customer_id;
            cout << "Password  : ";
            cin >> password;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            ifstream member_file("User_Details.txt");
            string line;
            while (getline(member_file, line))
            {
                User user;
                user.id = line;
                getline(member_file, user.password);
                getline(member_file, user.name);
                getline(member_file, user.nric);
                getline(member_file, user.email);
                getline(member_file, user.phone);
                getline(member_file, user.membership_level);

                if (customer_id == user.id && password == user.password)
                {
                    membership = user.membership_level;
                    if (membership == "gold" || membership == "Gold")
                    {
                        cout << "The customer is " << membership << " member. 5%
discount is given." << endl;
                        return 0.05;
                    }
                    else if (membership == "platinum" || membership == "Platinum")
                    {
                        cout << "The customer is " << membership << " member. 10%
discount is given." << endl;
                        return 0.1;
                    }
                    else if (membership == "diamond" || membership == "Diamond")
                    {
                        cout << "The customer is " << membership << " member. 15%
discount is given." << endl;
```

```
                        return 0.15;
                    }
                }
                else
                {
                    valid_member = false;
                }
            }
            member_file.close();
            if (!valid_member)
            {
                cout << "This customer is not a member." << endl;
                a = true;
            }
        }
        else
        {
            cout << "No discount is given." << endl;
            return 0.0;
        }
    }
}

double payment_method(double total_price_after_round)//payment method
{
    bool a = true, b = true;
    double  find_change = 0;
    char reset_payment;

    while (a)
    {
        cout << "What is the payment method of the customer? <E>-wallet / <C>ash /
<C>redit <C>ard : ";
        cin >> method;
        // the online payment typically is fit the amount of total price so no need
to find the change
        if (method == "E")
        {
            method = "E-wallet";
            find_change = 0;
            b = true;
        }
        else if (method == "C")
        {
            method = "Cash";
            cout << "Amount of the customer payed : ";
            cin >> customer_pay;
```

```
            if (!customer_pay)
            {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Invalid Input. Please pay again." << endl;
                a = true;
                b = false;
            }
            else if (customer_pay)
            {
                find_change = customer_pay - total_price_after_round;
                b = true;
            }
            else
            {
                cout << "Not allowed to key in the negative amount. Please pay
again." << endl;
                a = true;
                b = false;
            }
        }
        else if (method == "CC")
        {
            method = "Credit Card";
            find_change = 0;
            b = true;
        }
        else
        {
            cout << "This method is unacceptable and unsupported this system." <<
endl;
            a = true;
            b = false;
        }

        if (find_change < 0 && method == "Cash")//when the input amount of cash
unsufficient
        {
            cout << "The customer is paid by insufficient cash. Please pay again." <<
endl;
            a = true;
            b = false;
        }

        while (b)
        {
             cout << "Is the payment done? <Y>es / key in any key to reset the payment
method : ";
```

```
            cin >> reset_payment;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            if (toupper(reset_payment) == 'Y')
            {
                cout << "==================================" << endl;
                cout << "|| Find change (RM)|| " << right << setw(11) <<
setprecision(2) << fixed << find_change << "||" << endl;
                cout << "==================================" << endl;
                b = false;
                a = false;
            }
            else
            {
                b = false;
                a = true;
            }
        }

    }
    return find_change;
}

void history_list()
{
    int number = 0;
    cout << "History Record" << endl;
    cout <<
"================================================================================" <<
endl;
    cout << "||  Transaction_id  |  Customer Pay(RM)  |              Method
||" << endl;
    cout << "--------------------------------------------------------------------------------------------------------------------
---" << endl;
    ifstream myfile("history.txt");
    if (myfile.is_open())
    {
        string line, id, price, payment_method;

        while (myfile >> id)
        {
            myfile >> price;
            getline(myfile, payment_method);
            cout << "||       " << left << setw(11) << id << "|     " << setw(15) <<
price << "| " << setw(14) << payment_method << "     ||" << endl;
        }
        myfile.close();
```

```
        cout <<
"================================================================================" << endl
<< endl << endl;
    }
    else {
        cout << "Not recorded found!!\n" << endl;
    }

}


bool history()// search module
{
    bool found = true;
    string filename, transaction_id_search;
    do
    {
        system("cls");
        cout <<
"\t\t=============================================================================
====================================\n";
        cout << "\t\t||
||\n";
        cout << "\t\t||
||\n";
        cout << "\t\t||                    _     _   ________   _______   ________
____      ______    __    __                  ||\n";
        cout << "\t\t||                | |   | |  |___  __|  | _____|     |  __|  /
___ \\   |  ____\\   \\ \\  / /                    ||\n";
        cout << "\t\t||                | |___| |     | |     | |_____     | |    |
/  \\  |  | |___| |   \\ \\/ /                     ||\n";
        cout << "\t\t||                |  ____ |     | |     |______|     | |    |
|    | |  |  _  _/     |  |                     ||\n";
        cout << "\t\t||                | |   | |   __| |__    ____| |     | |    |
\\__/  |  | | \\ \\      |  |                     ||\n";
        cout << "\t\t||                |_|   |_|  |_______|  |______|     |_|
\\ _____/   |_|  \\_\\     |__|                      ||\n";
        cout << "\t\t||
||\n";
        cout << "\t\t||
||\n";
        cout << "\t\t||
||\n";
        cout <<
"\t\t=============================================================================
====================================\n";
        cout << endl << endl;
```

```
        history_list();

        cout << "Transaction ID (four digits number): ";
        cin >> transaction_id_search;
        cout << endl;

        if (isdigit(transaction_id_search[0]) && isdigit(transaction_id_search[1]) &&
isdigit(transaction_id_search[2]) && isdigit(transaction_id_search[2]) &&
transaction_id_search.length() == 4)
        {
            filename = "transacrtion_id_" + transaction_id_search + ".txt";
            ifstream myfile(filename);
            if (myfile.is_open())
            {
                string line;
                while (getline(myfile, line)) {
                    cout << line << endl;
                    found = true;
                }
                myfile.close();
            }
            else
                cout << "Transaction record not found.\n" << endl;
        }
        else
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid Input. Please only key in four digits number.\n" <<
endl;
        }

        if (!exit_page())
            transaction_id_search = "Q";

    } while (transaction_id_search != "Q");
    return true;
}

bool option4()
{
    bool continue_program = true;

    while (continue_program)
    {
        membership_interface();
        cout << "\n\t\t\t\t\t\t\t\t\t Option : "; // prompt user to enter option
```

```
        int menu_option;
        cin >> menu_option;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        if (menu_option == 1)
        {
            acc_registration();
            clear_screen();
        }
        else if (menu_option == 2)
        {
            acc_management();
            clear_screen();
        }
        else if (menu_option == 3)
        {
            continue_program = exit_page();
        }
        else
        {
            cout << "\t\t\tInvalid option! Please enter a number between 1 and 3." <<
endl;
        }
    }
    return true;
}

bool validate_user_id(string user_id)
{
    if (user_id.length() < 6 || user_id.length() > 12)   // Check if user ID is
within 6 to 12 characters
    {
        cout << "\t\tUser ID should be between 6 and 12 characters." << endl;   //
Notify the user that the ID should be within the specified range
        return false;
    }
    return true;
}

bool validate_password(string password)
{
    if (password.length() < 8 || password.length() > 20)  // Check if password is
within 8 to 20 characters
    {
        cout << "\t\tPassword should be between 8 and 20 characters." << endl;  //
Notify the user that the password should be within the specified range
        return false;
```

```
    }
    return true;
}

bool validate_confirm_password(string password, string confirm_password)
{
    if (password != confirm_password)     // Check if password same with
confirm_password
    {
        cout << "\t\tPassword confirmation does not match password." << endl;   //
Notify the user that the password and confirm password should be the same
        return false;
    }
    return true;
}

bool validate_full_name(string full_name)
{
    if (full_name.length() < 2 || full_name.length() > 50)    // Check if the length
of name is within 2 to 50 characters
    {
        cout << "\t\tFull name should be between 2 and 50 characters." << endl;    //
Notify the user that the length of name is within 2 to 50 characters
        return false;
    }
    return true;
}

bool validate_nric(string nric)
{
    // Check if the length of nric is 12 and all the characters are digits
    if (nric.length() == 12 && isdigit(nric[0]) && isdigit(nric[1]) &&
isdigit(nric[2]) && isdigit(nric[3]) && isdigit(nric[4]) && isdigit(nric[5]) &&
isdigit(nric[6]) && isdigit(nric[7]) && isdigit(nric[8]) && isdigit(nric[9]) &&
isdigit(nric[10]) && isdigit(nric[11]))
    {
        return true;
    }
    else
    {
        cout << "\t\tInvalid Format! Please try again!" << endl;  // Notify the user
that the format of nric is wrong
        return false;
    }
}

bool validate_email(string email)
{
```

```
    int at_symbol = email.find('@');
    int dot = email.find('.');

    if (at_symbol == string::npos || dot == string::npos || dot < at_symbol ||
email.find('@', at_symbol + 1) != string::npos || email.find('.', dot + 1) !=
string::npos)
    {
        cout << "\t\tInvalid email address. Please try again!" << endl;  // Notify
the user that the format of email address is wrong
        return false;
    }

    for (int i = 0; i < email.length(); i++)
    {
        if (!isalnum(email[i]) && email[i] != '.' && email[i] != '-' && email[i] !=
'_' && email[i] != '@') {
            cout << "\t\tInvalid email address. Please try again!" << endl;
            return false;
        }
    }
}

bool validate_phone_number(string phone_number)
{
    if (phone_number.length() < 9 || phone_number.length() > 11)   // Check if the
length of phone number is within 9 to 11 characters
    {
        cout << "\t\tPhone number should be between 9 and 11 digits." << endl;
        return false;
    }
    for (int i = 0; i < phone_number.length(); i++)
    {
        if (!isdigit(phone_number[i]))
        {
            cout << "\t\tPhone number should only contain digits." << endl;   //
Notify the user that the format of phone number is wrong
            return false;
        }
    }
    return true;
}

bool validate_membership_level(string membership_level)   // check if membership
level is valid
{
    if (membership_level == "Gold" || membership_level == "gold" || membership_level
== "Platinum" || membership_level == "platinum" || membership_level == "Diamond" ||
membership_level == "diamond")
```

```
    {
        return true;
    }
    else
    {
        cout << "\t\tInvalid input! Please try again!
(Gold/gold/Platinum/platinum/Diamond/diamond)" << endl;
        return false;
    }
}

bool id_used(string user_id)   // Check if the user ID is already used by another
user in the User_Details.txt file
{
    ifstream file("User_Details.txt");
    string line;
    while (getline(file, line))
    {
        if (line == user_id)
        {
            file.close();
            return true;
        }
    }
    file.close();
    return false;
}

bool nric_used(string user_nric)    // Check if the user nric is already used by
another user in the User_Details.txt file
{
    ifstream file("User_Details.txt");
    string line;
    while (getline(file, line))
    {
        if (line == user_nric)
        {
            file.close();
            return true;
        }
    }
    file.close();
    return false;
}

bool email_used(string user_email)    // Check if the user email is already used by
another user in the User_Details.txt file
```

```
{
    ifstream file("User_Details.txt");
    string line;
    while (getline(file, line))
    {
        if (line == user_email)
        {
            file.close();
            return true;
        }
    }
    file.close();
    return false;
}

bool phone_used(string user_phone)    // Check if the user phone number is already
used by another user in the User_Details.txt file
{
    ifstream file("User_Details.txt");
    string line;
    while (getline(file, line))
    {
        if (line == user_phone)
        {
            file.close();
            return true;
        }
    }
    file.close();
    return false;
}

void membership_interface()
{
    system("cls");
    cout <<
"\t\t=============================================================================
================================================\n";
    cout << "\t\t||
||\n";
    cout << "\t\t||
||\n";
    cout << "\t\t||         ___    ___    ______    ___    ___    ______     ______
______     ______    _     _    ________   _______         ||\n";
    cout << "\t\t||        |   \\  /   |  |  ____|  |   \\  /   |  |  ____\\   |
____|  |  ____\\   | _____|  | |   | |  |__  __|  |  ___\\\        ||\n";
    cout << "\t\t||        | |\\ \\/ /| |  | |____   | |\\ \\/ /| |  | |___| |  |
|____   | |___| |  | |_____  | |__| |     | |     | |___| |       ||\n";
```

```
    cout << "\t\t||        | | \\  / | |  |  ___|   | | \\  / | |  |  ____/   |  ___|
|  _  _/   |______|  |  ____ |     | |     |  ____/        ||\n";
    cout << "\t\t||        | |  \\/  | |  | |_____  | |  \\/  | |  | |___| \\  |
|_____  | | \\ \\     ____| |  | |   | |      __| |  | |             ||\n";
    cout << "\t\t||        |_|      |_|  |______|  |_|      |_|  |_______/  |______|
|_|  \\_\\   |______|  |_|   |_|  |_______|  |_|             ||\n";
    cout << "\t\t||
||\n";
    cout << "\t\t||
||\n";
    cout << "\t\t||
||\n";
    cout <<
"\t\t=============================================================================
================================================\n";
    cout << "\n\n\n";
    cout << "\t\t======================================
\t====================================== \t==================================\n";
    cout << "\t\t|| 1 ||      Account Registration     || \t|| 2 ||      Account
Management       || \t|| 3 ||      Exit               ||\n";
    cout << "\t\t======================================
\t====================================== \t==================================\n";
    cout << "\n" << endl;

}

void acc_registration()//registration module
{
    string user_id, password, confirm_password, full_name, nric, email, phone_number,
membership_level;
    bool is_valid, valid_level = false;

    clear_screen();
    cout <<
"\t\t=============================================================================
==============" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t||                    Welcome to KLN Cinema Membership
Registration!                    ||" << endl;
    cout <<
"\t\t||_____________________________________________________________________________
_____________||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Register to become a member and receive the following benefits:
||" << endl;
    cout << "\t\t||
||" << endl;
```

```
    cout << "\t\t|| - Get the latest movie information by email ~
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| - Enjoy different discount rates for the payment ~
||" << endl;
    cout << "\t\t||
||" << endl;
    cout <<
"\t\t================================================================================
==============" << endl;
    cout << "\t\t|| ID:
||" << endl;
    cout << "\t\t|| ( Must be within 6-12 characters )
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Password:
||" << endl;
    cout << "\t\t|| ( Must be within 8-20 characters )
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Confirm Password:
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Full Name:
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| NRIC:
||" << endl;
    cout << "\t\t|| ( Please use only 12 digit NRIC number )
||" << endl;
    cout << "\t\t|| ( Example: 97XXXX0X1XXX )
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Email:
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Phone Number:
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Membership Level:
||" << endl;
```

```
    cout << "\t\t|| 1. Gold     ( 5% discount rate at total payment )
||" << endl;
    cout << "\t\t|| 2. Platinum ( 10% discount rate at total payment )
||" << endl;
    cout << "\t\t|| 3. Diamond  ( 15% discount rate at total payment )
||" << endl;
    cout <<
"\t\t=================================================================================
==============" << endl;

    while (true)
    {
        cout << "\t\tDo you want to register for our membership? (Y/N) :";  // Ask
the user if they want to register for membership
        string option;
        getline(cin, option);

        if (option == "n" || option == "N")    // If the user enters 'n' or 'N', exit
the function
        {
            return;
        }
        else if (option == "y" || option == "Y")  // If the user enters 'y' or 'Y',
break out of the loop and continue with registration
        {
            break;
        }
        else   // If the user enters anything else, notify them that their input was
invalid and ask again
        {
            cout << "\t\tInvalid input! Please try again." << endl;
        }
    }
    do
    {
        cout << "\t\tID:";
        getline(cin, user_id);
        if (id_used(user_id))
        {
            cout << "\t\tSorry, this ID has been taken by others. Please enter a new
one." << endl; // Notify the user about the length requirement for user ID
            is_valid = false;
        }
        else
        {
            is_valid = validate_user_id(user_id);
        }
    } while (!is_valid);
```

```
    do
    {
        cout << "\t\tPassword:";
        getline(cin, password);
        is_valid = validate_password(password);    // Validate the password
    } while (!is_valid);

    do
    {
        cout << "\t\tConfirm Password:";
        getline(cin, confirm_password);
        is_valid = validate_confirm_password(password, confirm_password);    //
Validate the confirm password
    } while (!is_valid);

    do
    {
        cout << "\t\tFull Name:";
        getline(cin, full_name);
        is_valid = validate_full_name(full_name);  // Validate the full name
    } while (!is_valid);

    do
    {
        cout << "\t\tNRIC:";
        getline(cin, nric);
        if (nric_used(nric))
        {
            cout << "\t\tSorry, this NRIC has been taken by others. Please enter a
new one." << endl; // Notify the user about the NRIC requirements
            is_valid = false;
        }
        else
        {
            is_valid = validate_nric(nric);    // Validate the nric
        }

    } while (!is_valid);

    do
    {
        cout << "\t\tEmail:";
        getline(cin, email);
        if (email_used(email))
        {
```

```
            cout << "\t\tSorry, this email has been taken by others. Please enter a
new one." << endl; // Notify the user about the email requirements
            is_valid = false;
        }
        else
        {
            is_valid = validate_email(email);    // Validate the email
        }
    } while (!is_valid);

    do
    {
        cout << "\t\tPhone Number:";
        getline(cin, phone_number);
        if (phone_used(phone_number))
        {
            cout << "\t\tSorry, this phone number has been taken by others. Please
enter a new one." << endl;  // Notify the user about the phone number requirements
            is_valid = false;
        }
        else
        {
            is_valid = validate_phone_number(phone_number); // Validate the phone
number
        }
    } while (!is_valid);

    do
    {
        cout << "\t\tMembership Level (Gold/Platinum/Diamond):";
        getline(cin, membership_level);
        is_valid = validate_membership_level(membership_level);    // Validate the
membership level
    } while (!is_valid);


    ofstream file("User_Details.txt", ios::app);  // Open the file in append mode
    if (file.is_open())
    {
        // Write user details to the file
        file << user_id << endl;
        file << password << endl;
        file << full_name << endl;
        file << nric << endl;
        file << email << endl;
        file << phone_number << endl;
        file << membership_level << endl;
```

```
        file.close();
    }
    else
    {
        cout << "\t\tUnable to open file!" << endl;
    }
    // Display a success message and pause the program
    cout << "\t\tThank you for registering our membership! Your user details are
recorded." << endl;
    system("Pause");
    return;
}

void acc_management()//update module
{
    clear_screen();
    cout <<
"\t\t================================================================================
==============" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t||                        Welcome to KLN Cinema Membership Management!
||" << endl;
    cout <<
"\t\t||_____________________________________________________________________________
_____________||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| Menu of options:
||" << endl;
    cout << "\t\t||
||" << endl;
    cout << "\t\t|| 1. Update name
||" << endl;
    cout << "\t\t|| 2. Update NRIC
||" << endl;
    cout << "\t\t|| 3. Update email
||" << endl;
    cout << "\t\t|| 4. Update phone number
||" << endl;
    cout << "\t\t|| 5. Update membership level
||" << endl;
    cout << "\t\t|| 6. Save changes and exit
||" << endl;
    cout << "\t\t||
||" << endl;
    cout <<
"\t\t================================================================================
==============" << endl;
```

```
    while (true)
    {
        cout << "\t\tDo you want to register for our membership? (Y/N) :";  // Ask
the user if they want to register for membership
        string option;
        getline(cin, option);

        if (option == "n" || option == "N")    // If the user enters 'n' or 'N', exit
the function
        {
            return;
        }
        else if (option == "y" || option == "Y")  // If the user enters 'y' or 'Y',
break out of the loop and continue with registration
        {
            break;
        }
        else   // If the user enters anything else, notify them that their input was
invalid and ask again
        {
            cout << "\t\tInvalid input! Please try again." << endl;
        }
    }

    bool loggedIn = false;//log in module

    ifstream file("User_Details.txt");

    if (file.is_open())                      // Check if the file is open
    {
        cout << "\n";
        cout << "\t\tPlease log in to update your details." << endl;
        string id, password;
        cout << "\t\tID:";
        getline(cin, id);
        cout << "\t\tPassword:";
        getline(cin, password);

        string line;
        ofstream temp("temp.txt");       // Create a temporary file to store updated
user details

        while (getline(file, line))     // Loop through each line in the file and
read the user details
        {
```

```
            User user;                    // Create a new User object to hold the
details
            user.id = line;
            getline(file, user.password);
            getline(file, user.name);
            getline(file, user.nric);
            getline(file, user.email);
            getline(file, user.phone);
            getline(file, user.membership_level);

            if (id == user.id && password == user.password)     // Check if the
entered ID and password match those in the text file
            {
                cout << "\n";
                cout << "\t\tWelcome," << user.name << "!" << endl;
                cout << "\t\tThis is your membership details:" << endl;
                cout << "\t\tNRIC:" << user.nric << endl;
                cout << "\t\tEmail:" << user.email << endl;
                cout << "\t\tPhone number:" << user.phone << endl;
                cout << "\t\tMembership Level:" << user.membership_level << endl;

                loggedIn = true;

                string new_value;
                cout << "\n";
                cout << "\t\tEnter new value for each field (or press Enter to keep
current value):" << endl;

                // Prompt user for new values for each field and update user object
accordingly
                cout << "\t\tFull Name: ";
                getline(cin, new_value);
                if (!new_value.empty())
                {
                    if (validate_full_name(new_value))
                    {
                        user.name = new_value;
                    }
                }

                cout << "\t\tNRIC: ";
                getline(cin, new_value);
                if (!new_value.empty())
                {
                    if (validate_nric(new_value))
                    {
                        user.nric = new_value;
```

```
                    }
                }
                cout << "\t\tEmail: ";
                getline(cin, new_value);
                if (!new_value.empty())
                {
                    if (validate_email(new_value))
                    {
                        user.email = new_value;
                    }
                }

                cout << "\t\tPhone Number: ";
                getline(cin, new_value);
                if (!new_value.empty())
                {
                    if (validate_phone_number(new_value))
                    {
                        user.phone = new_value;
                    }
                }
                cout << "\t\tMembership Level: ";
                getline(cin, new_value);
                if (!new_value.empty())
                {
                    if (validate_membership_level(new_value))
                    {
                        user.membership_level = new_value;
                    }
                }

                cout << "\t\tYour details have been updated." << endl;
                system("Pause");

                temp << user.id << endl; // Write the updated user details to the
temporary file
                temp << user.password << endl;
                temp << user.name << endl;
                temp << user.nric << endl;
                temp << user.email << endl;
                temp << user.phone << endl;
                temp << user.membership_level << endl;
            }
            else
            {
                temp << line << endl; // Write the details of other users to the
temporary file
```

```
                temp << user.password << endl;
                temp << user.name << endl;
                temp << user.nric << endl;
                temp << user.email << endl;
                temp << user.phone << endl;
                temp << user.membership_level << endl;
            }
        }

        file.close();
        temp.close();

        remove("User_Details.txt");                // Delete the old file
        rename("temp.txt", "User_Details.txt");   // Rename the temporary file to the
original file name
    }
    if (!loggedIn)  // If the ID and password do not match, deny access
    {
        cout << "\t\tAccess denied." << endl;
        system("Pause");
        return;
    }
}

void clear_screen()
{
    system("cls");
}
```