Library Management System

- library management system is a small project
- used to store book information
- uses file handling
- creates books.txt file
- stores the book information in the books.txt file

Features of Library Management System

Display books

- creates book.txt if not found
- if found displays books in table

Add books

- return error if book.txt is not found so you need to display books first

Remove books

- remove book by name
- remove book by id

Clear the screen

Quit

** This project assigns a unique id to each book but not serial not

---

Algorithm of Library Management System

---

before running program

- include libraries
- define files
- define structures
- protype the function

Main

- say welcome
- display main menu

---

main menu

- show options {

1. display books
2. add books
3. remove books
4. clear
5. quit } -if 1,2,3,4,5 then display books, add books, remove books, clear, quit -else say error

---

display books

- checkDB
- if there is DB then display books by skipping the first line
- else if there is no DB then createDB

createDB

- make a file named books.txt namely books
- ask for the capacity of library and add it to the first line
- show the books by skipping the first line that contains capacity

---

add books

- open books in append mode
- if checkDB return 1 then append books
- else createDB then append books

append books

- open books in append mode
- get the last id from the books.txt

get last id

- skip first line that has capacity
- iterate upto last line while putting the value for last id(the last iteration will give the last id)
- return lastid
- ask for book name, book author
- this books id = lastid + 1
- put these information in the file

---

remove book

-open books in read mode

- open temp file in write mode
- ask if user want to remove by id or name
- if by name them remove book by name
- if by id then remove book by id

---

remove book by id

- open books in read mode
- open temp file in write mode
- copy capacity of book to temp file
- iterate{
    - check if the id of the book mathches the id specified by the user
    - if not matches, write the book to the temp file
    - if matches, dont write the book to the temp file
- remove the books.txt
- rename the temp.txt to books.txt } -close books

---

remove by name

- open books in read mode
- open temp file in write mode
- copy capacity of book to temp file
- iterate{
    - check if the name of the book mathches the name specified by the user
    - if not matches, write the book to the temp file
    - if matches, dont write the book to the temp file
- remove the books.txt
- rename the temp.txt to books.txt } -close books

---

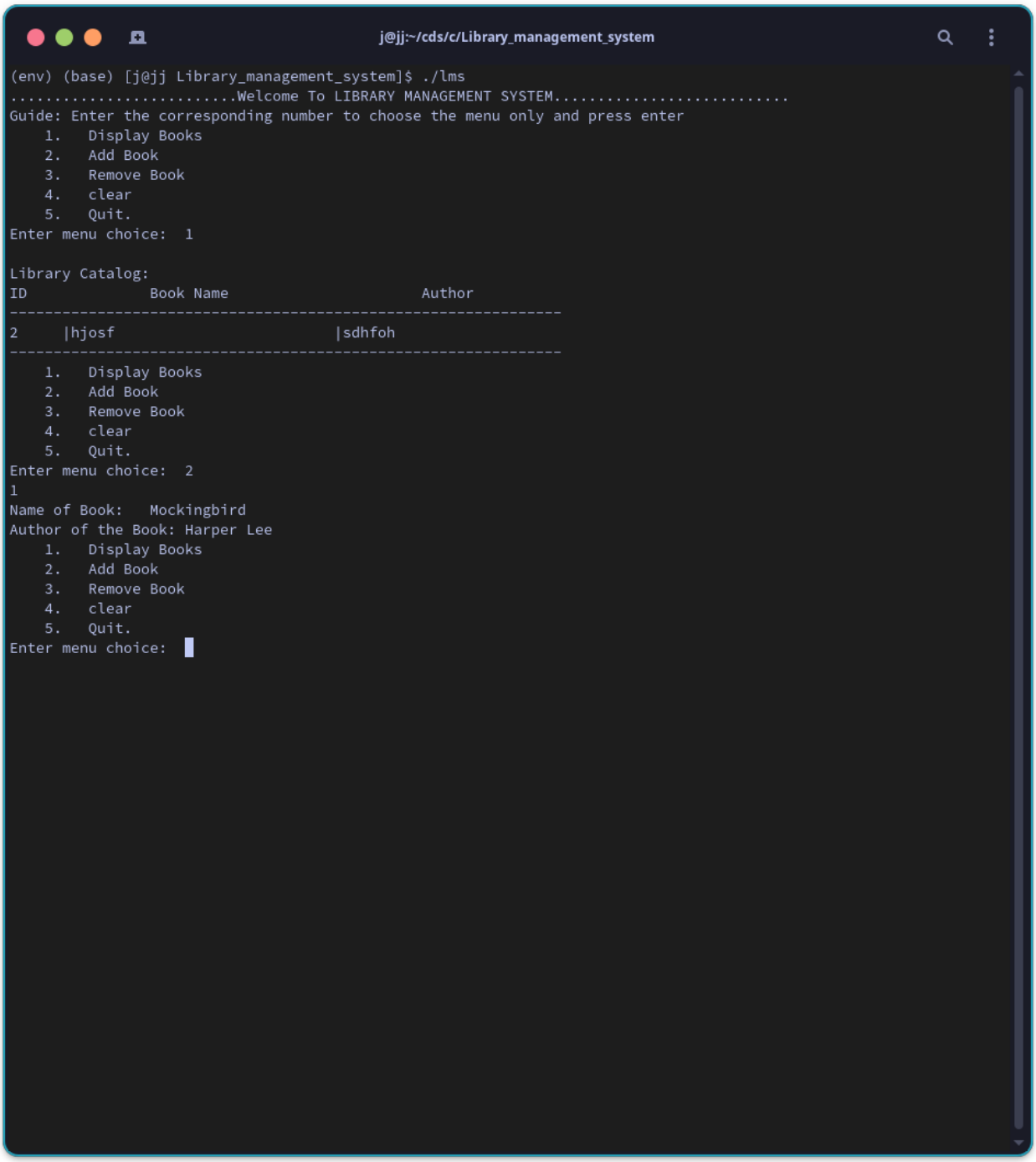
clear

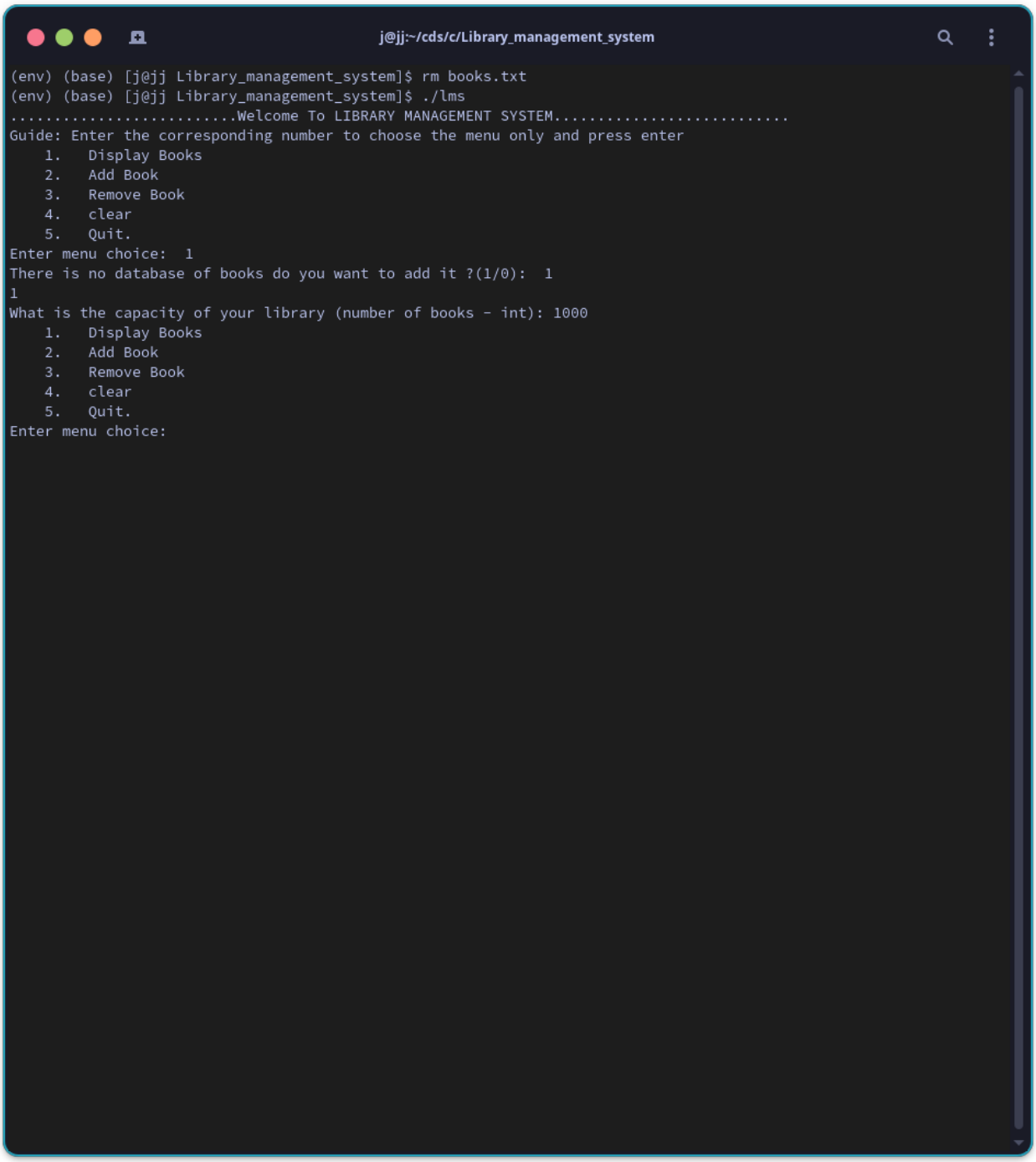
- tell system terminal (bash) to clear the screen

quit

- quit the program

Screenshot


j@jj:~/cds/c/Library_management_system
(env) (base) [j@jj Library_management_system]$ ./lms
.....Welcome To LIBRARY MANAGEMENT SYSTEM.....
Guide: Enter the corresponding number to choose the menu only and press enter
1. Display Books
2. Add Book
3. Remove Book
4. clear
5. Quit.
Enter menu choice: 1
Library Catalog:
ID Book Name Author
2 |hjosf |sdhfoh
1. Display Books
2. Add Book
3. Remove Book
4. clear
5. Quit.
Enter menu choice: 2
1
Name of Book: Mockingbird
Author of the Book: Harper Lee
1. Display Books
2. Add Book
3. Remove Book
4. clear
5. Quit.
Enter menu choice:



Billing Management System

- Uses structures
- phone billing System

Features

- show the total bill
- add bill
- remove bill
- calculate cost for minutes of call

Algorithm for Billing Management System

Algorithm for Operations

---

1. Adding a New Record (`addRecord` Function)

- Prompts the user to input customer information (name, phone number, and usage in minutes).
- Calculates the total bill based on usage.
- Adds the new record to the `customers` array.

2. Viewing List of Records (`viewRecords` Function)

- Prints a table header.
- Iterates through the `customers` array and prints each customer's information.

3. Modifying a Record (`modifyRecord` Function)

- Takes a phone number as input to identify the record to be modified.
- Prompts the user to enter the new usage in minutes.
- Updates the total bill based on the new usage.

4. Viewing Payment (`viewPayment` Function)

- Takes a phone number as input to identify the record to be viewed.
- Prints the name, phone number, usage, and total bill for the specified customer.

5. Deleting a Record (`deleteRecord` Function)

- Takes a phone number as input to identify the record to be deleted.
- Shifts the remaining records to fill the gap created by the deleted record.

6. Displaying Menu and Main Loop (`displayMenu` and `main` Functions)

- Displays a menu with options for different operations.
- Takes user input to execute the chosen operation.
- The program continues to run until the user chooses to exit.

Screenshot of Billing Management System

```
ALGORITHM.docx  billing-system  billing-system.c  billing-system.exe  README.md
(env) (base) [j@jj Billing_management_system]$ ./billing-system

1. Add New Record
2. View List of Records
3. Modify Record
4. View Payment
5. Delete Record
6. Exit
Enter your choice: 1

Enter name: Hello
Enter phone number: 989898989
Enter usage (in minutes): 600

Record added successfully!

1. Add New Record
2. View List of Records
3. Modify Record
4. View Payment
5. Delete Record
6. Exit
Enter your choice: 2

Name                Phone Number        Usage(min)     Total Bill($)
Hello               989898989           600.00         60.00

1. Add New Record
2. View List of Records
3. Modify Record
4. View Payment
5. Delete Record
6. Exit
Enter your choice: 4

Enter phone number to view payment: 989898989
Hello   989898989       600.00          60.00

1. Add New Record
2. View List of Records
3. Modify Record
4. View Payment
5. Delete Record
6. Exit
Enter your choice: 5

Enter phone number to delete record: 989898989

Record deleted successfully!
```

Hangman game

- simple hangman game

Hangman Game Algorithm

Initialization:

- Import necessary libraries (`stdio.h`, `stdlib.h`, `string.h`, `ctype.h`, `time.h`).
- Declare function prototypes for `drawHangman` and `strlwr`.
- Define `main` function:
  - Declare variables and arrays for the game: - `choice` for user input to play again. - `guessWords` array containing words for the game. - `randomIndex` to randomly select a word index. - `numLives` for tracking remaining lives. - `numCorrect` for counting correct guesses. - `oldCorrect` to compare with `numCorrect` for wrong guesses. - `lengthOfWord` for the length of the selected word. - `letterGuessed` array to track guessed letters. - `quit` to determine if the user quit early. - `loopIndex` for iterating over the word. - `reguessed` to check if a letter has been guessed before. - `guess` array to store user input. - `letterEntered` to store the guessed letter.

Game Loop:

- Start a do-while loop to allow multiple game plays.
  - Seed the random number generator with `srand(time(NULL))`.
- Randomly select a word index (`randomIndex`) from `guessWords`.
- Initialize game variables (`numLives`, `numCorrect`, `oldCorrect`, etc.).
- Start a while loop for each turn of the game until the word is guessed or lives run out.
  - Display the current state of the word with underscores for unguessed letters.
  - Display the hangman figure based on the number of incorrect guesses.
  - Prompt the player for a letter guess.
  - Convert the guess to lowercase using the `strlwr` function.
  - Check if the player wants to quit (`strncmp(guess, "quit", 4)`).
  - Clear the console screen (`system("clear")` or `system("cls")`).
  - Process the user's guess:
  - Check if the letter has already been guessed.
  - Update the game state based on the guess.

    - Check if the player has won or lost.
    - If the player wants to quit, exit the loop.
- Display the game result (win, lose, or quit).
- Ask the player if they want to play again (`printf("\nDo you want to Play Again? (Y/N)\n")`).
- Read the player's choice into the `choice` variable.
- Continue the loop if the player wants to play again (`while (choice == 'Y' || choice == 'y')`).

Drawing the Hangman:

- Implement the `drawHangman` function to display the hangman figure based on the number of lives.

Convert String to Lowercase:

- Implement the `strlwr` function to convert a string to lowercase.

End of Program:

- Return 0 from the `main` function.

Screenshot for Hangman

```
Letter Entered:d
Lives used:3/5
Correct guess :)


New Turn....
Hangman Word:___d___
     _________
    |         |
    |         O

Number Correct So Far:1
Enter a guess letter:
```

Quiz game

- play a simple quiz game
- uses file handling
- creates qna.txt file
- stores the questions and answers in the qna.txt file

Features of Quiz game

- quiz game with questions and answers
- display questions
- display answers
- display score
- save score

Algorithm of Quiz game

---

before running the function

- include libraries of string ctypes stdio stdlib
- include qna.txt which contains question,4 options,corresponding_correct_answer_no
- define functions tolower

toLowercase *str(call by reference)

- initialize i = 0
- iterate over the string
    - lower the character

---

get_no_of_question

- open qna
- init i = 0
- while seeing question, 4 options, and the correct answer increase i
- print no of question
- return no of question

MAIN FUCTION

- init score = 0, noofq = get_no_of_questions
- open qna
- loop init i = 0 upto less than no of questions with increment of one question
    - see i th line of qna
    - print questions then options

    - take user input as answer
    - if answer == correct option increase score
    - else go to next question
- float percentage = score/noofq * 100
- if percentage == 100 >> perfect score
- else if >= 80 >> excellent score out of noofq
- else if >= 60 >> good score out of noofq
- else if >= 40 >> satisfactory score out of noofq
- else if < 40 >> failed score out of noofq
- else >> something went wrong
- return 0

Screenshot for quiz

```
(env) (base) [j@jj c]$ ls
Billing_management_system  Library_management_system  quiz       tictactoe
Hangman                    Presentation               README.md
(env) (base) [j@jj c]$ cd quiz
(env) (base) [j@jj quiz]$ ls
note.txt  qna.txt  quiz  quiz.c  quiz.exe  README.md
(env) (base) [j@jj quiz]$ ./quiz
Total number of questions: 30
Who invented the telephone?
1. Thomas Edison
2. Alexander Graham Bell
3. Nikola Tesla
4. Guglielmo Marconi
Enter your answer (1, 2, 3, or 4): 2
Hmm....
What is the national bird of the United States?
1. Peregrine Falcon
2. Bald Eagle
3. Blue Jay
4. Cardinal
Enter your answer (1, 2, 3, or 4): 2
```

Tic Tac Toe Game

- uses 2D array to display the game
- uses indexing to input the position of player X or 0

Algorithm of Tic Tac Toe Game

Initialization

- Define a 3x3 character array `board` to represent the Tic Tac Toe board.
- Implement the `initializeBoard` function to initialize the board with empty spaces and display cell numbers.
- Create a main function (`main`) to set up the game, display instructions, and start the game loop.

Display Functions

`initializeBoard` Function

- Initialize the `board` array with empty spaces.
- Display the initial board layout with numbered cells.

`showBoard` Function

- Display the current state of the Tic Tac Toe board.

Update Board

`updateBoard` Function

- Take a cell number and player's sign as input.
- Calculate the corresponding row and column for the cell number.
- Check if the selected cell is already filled; if not, update the board with the player's sign.
- Display the updated board.

Check Winner

`checkWinner` Function

- Check for winning conditions in rows, columns, and diagonals.
- Return 1 if a player has won; otherwise, return 0.

Game Loop

`playTicTacToe` Function

- Initialize variables for game results, selected cell, play count, and update result.
- Alternate turns between Player 1 (X) and Player 2 (O).
- Accept user input for cell selection.
- Update the board and check for a winner after each move.
- Display the result at the end of the game.

Main Function

- Display game instructions and initial board.
- Wait for user input to start the game.
- Start the game loop.
- Provide a menu for restarting or exiting the game.
- Continue playing or exit based on user choice.

Termination

- Thank the player for participating.

Execution

- Compile and run the program.
- Follow on-screen instructions to play Tic Tac Toe.
- Terminate the game or restart based on user input.

Screenshot for Tic Tac Toe

```
(env) (base) [j@jj c]$ ls
Billing_management_system  Library_management_system  quiz       tictactoe
Hangman                    Presentation               README.md
(env) (base) [j@jj c]$ cd tictactoe
(env) (base) [j@jj tictactoe]$ ls
README.md  tictactoe  tictactoe.c
(env) (base) [j@jj tictactoe]$ ls
README.md  tictactoe  tictactoe.c
(env) (base) [j@jj tictactoe]$ ./tictactoe
------Tic Tac Toe------


 * Instruction

        Player 1 sign=X
        Player 2 sign=O
         To exit from game,Enter -1


* Cell Number on Board


         1  |  2  |  3
        ----------------
         4  |  5  |  6
        ----------------
         7  |  8  |  9


>>>>> Press Enter to start ...

 Player 1 [X]:5


            |     |
        ----------------
           |  X  |
        ----------------
           |     |



 Player 2 [O]:
```

Any Queries??