SINGLY LINKED LIST:

1. e-commerce website : You are building an online shopping cart system for an e-commerce website. The cart should efficiently manage the items added by users and calculate the total price of the items. Implement a singly linked list to keep track of the items in the cart. Each node in the linked list should contain the product details, such as name, price, and quantity. Write functions to add items to the cart, remove items by their name, and calculate the total price of all the items in the cart.
2. TEXT EDITOR: You are working on a text editor application that needs to handle large documents efficiently. Develop a singly linked list to store the lines of a document. Each node should represent a line of text, and the linked list should maintain the order of the lines. Implement functions to insert a new line at a specific position, delete a line by its line number, and search for a specific word within the document. Additionally, provide a function to count the total number of lines in the document.
3. SOCIAL MEDIA PALTFORM: You are creating a social media application where users can follow each other and build connections. Design a singly linked list to represent the followers of a user. Each node in the linked list should contain the user's details, such as name, profile picture, and follower count. Write functions to add followers, remove followers, and display the followers of a particular user. Additionally, provide a function to find the common followers between two users.
4. MOVIE TICKET: You are building a ticket booking system for a theater with a large seating capacity. Implement a singly linked list to manage the available seats. Each node in the linked list should represent a seat, and it should store information such as seat number, row, and availability status. Develop a function to find the best available seat for a new booking, considering factors like proximity to the center and grouping seats together. Write additional functions to reserve a seat, cancel a reservation, and display the seating arrangement.
5. AI ASSISTENT: You are working on a scheduling application for a busy professional. Create a singly linked list to store the appointments for a day. Each node in the linked list should represent an appointment and store details like start time, end time, and description. Implement functions to add appointments, delete appointments, and check for any overlap between two appointments. Additionally, provide a function to display the day's schedule in chronological order.
6. Delivery Service Navigation System : You are developing a navigation system for a delivery service that operates in a city with multiple stops. Design a singly linked list to represent the stops on a delivery route. Each node in the linked list should contain information about a stop, such as address, coordinates, and package details. Write functions to add stops to the route, remove stops, and calculate the total distance covered by the delivery route. Additionally, provide a function to optimize the route by rearranging the stops for better efficiency.
7. University Student Management System : You are building a student management system for a university. Implement a singly linked list to store the student records. Each node in the linked list should represent a student and contain information like name, ID, grades, and enrollment status. Develop functions to add students, remove students by their ID, and find the average grade of all the students. Additionally, provide a function to display the students' information in alphabetical order
8. DNA Sequence Analysis:

You are working on a DNA sequence analysis project where you need to process and analyze large sets of genetic data. Design a singly linked list to represent DNA sequences, where each node contains a nucleotide (A, T, C, or G) and a pointer to the next nucleotide in the sequence. Implement functions to concatenate two DNA sequences, search for specific patterns within the sequences, and calculate the percentage of GC content in a given sequence.

1. Cryptocurrency Transaction Ledger:

You are developing a blockchain-based cryptocurrency and need to maintain a ledger of all the transactions. Implement a singly linked list to represent the transaction history, where each node contains the sender's address, receiver's address, amount transferred, and a pointer to the next transaction. Write functions to add new transactions to the ledger, validate the integrity of the ledger using hashing algorithms, and calculate the total balance for a specific address.

1. Autonomous Vehicle Routing System:

You are part of a team building an autonomous vehicle routing system for a smart city. Design a singly linked list to represent the road network, where each node represents a road segment and contains information like start and end coordinates, traffic conditions, and maximum speed limit. Implement functions to find the shortest path between two locations, optimize routes based on real-time traffic updates, and detect road closures or accidents to reroute the autonomous vehicles efficiently.

DOUBLY LINKED LIST

Library Book Management System:

You are developing a book management system for a library. Design a doubly linked list to represent the collection of books. Each node should contain information about a book, such as title, author, and ISBN. Implement functions to add books to the collection, remove books by their ISBN, and display the books in alphabetical order by title. Additionally, provide a function to track the borrowing history of each book, including the borrower's details and return dates.

Image Editing Application:

You are working on an image editing application that requires efficient handling of multiple layers. Develop a doubly linked list to store the image layers, where each node represents a layer and contains information like layer name, opacity, and blending mode. Implement functions to add new layers, remove layers, and reorder the layers based on user input. Additionally, provide functions to apply various image manipulation operations on the layers, such as blending, masking, and adjusting transparency.

Train Reservation System:

You are building a train reservation system for a railway company. Design a doubly linked list to manage the passenger reservations for each train. Each node in the linked list should represent a reservation and store information like passenger name, seat number, and departure station. Write functions to add new reservations, cancel reservations by passenger name, and display the passenger details in the order of their reservation. Additionally, provide a function to find the number of available seats on a specific train.

Task Scheduler:

You are developing a task scheduling application for personal use. Create a doubly linked list to store the tasks, where each node represents a task and contains details such as task name, priority, and deadline. Implement functions to add new tasks, mark tasks as completed, and reorder the tasks based on priority and deadline. Additionally, provide a function to display the tasks in chronological order, highlighting overdue tasks.

Online Chess Game:

You are building an online multiplayer chess game platform. Design a doubly linked list to represent the game moves, where each node contains information about a move, including the piece moved, source, and destination squares. Implement functions to add new moves, undo moves, and validate the legality of each move based on the game rules. Additionally, provide functions to replay the game moves and display the current board position after each move.

Hotel Room Management System:

You are developing a hotel room management system for a luxury hotel. Design a doubly linked list to represent the rooms in the hotel. Each node should contain information about a room, such as room number, occupancy status, and guest details. Implement functions to add new rooms, mark rooms as occupied or vacant, and search for available rooms based on specific criteria like room type or amenities. Additionally, provide functions to check guests in and out of rooms and display the current room occupancy status.

Music Player with Playlist Navigation:

You are working on a music player application with advanced playlist navigation features. Create a doubly linked list to manage the playlists, where each node represents a playlist and stores information such as playlist name and song details. Implement functions to add new playlists, delete playlists, and navigate between playlists in both forward and backward directions. Additionally, provide functions to add songs to playlists, remove songs, and support seamless playback between songs in different playlists.

Airport Baggage Handling System:

You are designing a baggage handling system for an airport. Develop a doubly linked list to manage the flow of baggage from check-in to the boarding gate. Each node in the linked list should represent a baggage item and contain information like baggage ID, passenger details, and destination. Implement functions to add baggage items at check-in, track the movement of baggage between different checkpoints, and ensure accurate delivery of baggage to the appropriate boarding gate. Additionally, provide functions to handle exceptions like lost or mishandled baggage.

Vehicle Fleet Management:

You are building a vehicle fleet management system for a logistics company. Design a doubly linked list to represent the fleet of vehicles, where each node represents a vehicle and stores information such as vehicle number, driver details, and current location. Implement functions to add new vehicles, remove vehicles from the fleet, and track the movement of vehicles between different locations. Additionally, provide functions to calculate the distance traveled by each vehicle and optimize the route planning for efficient logistics operations.

Online Quiz Platform:

You are developing an online quiz platform for educational purposes. Create a doubly linked list to manage the quiz questions, where each node represents a question and contains information like question text, options, and correct answer. Implement functions to add new questions, delete questions, and allow users to navigate through the quiz in both forward and backward directions. Additionally, provide functions to score the quizzes, display the results, and provide feedback on the user's performance.

CIRCULAR INKED LIST:

Round-Robin Scheduling Algorithm:

You are developing an operating system scheduler that uses the round-robin scheduling algorithm. Implement a circular linked list to manage the list of processes waiting to be executed. Each node in the linked list represents a process and contains information such as process ID, execution time, and priority. Implement functions to add new processes, remove completed processes, and rotate the execution order based on a fixed time quantum. Additionally, provide functions to display the current scheduling order and handle priority adjustments.

Circular Music Playlist:

You are building a music player application with a circular playlist feature. Design a circular linked list to represent the songs in the playlist. Each node should contain information about a song, such as song name, artist, and duration. Implement functions to add new songs, remove songs, and allow seamless looping of the playlist. Additionally, provide functions to skip to the next song and display the current playing song.

Josephus Problem:

You are tasked with solving the Josephus problem, which involves a circle of people where every 'kth' person is eliminated until only one person remains. Implement a circular linked list to represent the circle of people. Each node should contain information about a person, such as their name or ID. Write functions to add people to the circle, eliminate people based on the elimination rule, and identify the last person remaining. Additionally, provide a function to display the order of elimination.

Circular Queue:

You are working on implementing a circular queue data structure for a ticketing system at a theme park. Design a circular linked list to manage the tickets in the queue. Each node should represent a ticket and contain information such as ticket ID, customer details, and ride preference. Implement functions to enqueue new tickets, dequeue tickets for processing, and handle the circular nature of the queue. Additionally, provide functions to display the current queue status and manage priority ticketing.

Circular Messaging System:

You are developing a messaging system for a group of users. Implement a circular linked list to manage the messages in a circular manner, ensuring that every user receives each message exactly once. Each node in the linked list should represent a message and contain information such as message content, sender, and receiver. Implement functions to add new messages, distribute messages in a circular manner, and handle message acknowledgments. Additionally, provide functions to display the message history and track message delivery status.

Circular Task Scheduler:

You are developing an embedded system that requires a task scheduler with a circular execution order. Implement a circular linked list to manage the tasks. Each node in the linked list represents a task and contains information such as task ID, priority, and execution time. Implement functions to add new tasks, remove completed tasks, and rotate the execution order based on task priorities. Additionally, provide functions to display the current task schedule and handle task dependencies.

Circular Buffer:

You are designing a data buffer for a real-time audio processing application. Implement a circular linked list to manage the audio samples in a circular buffer. Each node should represent an audio sample and contain information such as sample value, timestamp, and channel information. Implement functions to add new samples, retrieve samples for processing, and handle buffer wraparound. Additionally, provide functions to display the current buffer state and manage buffer overflow/underflow conditions.

Circular Railway System:

You are working on a railway system that operates in a circular loop, connecting multiple stations. Design a circular linked list to represent the stations on the railway route. Each node in the linked list should contain information about a station, such as station name, arrival/departure times, and platform details. Write functions to add new stations, remove stations, and calculate the total distance covered by the railway route. Additionally, provide a function to optimize the route by rearranging the stations for better efficiency.

Circular Job Queue:

You are developing a job queue for a distributed computing system. Implement a circular linked list to manage the jobs in the queue. Each node should represent a job and contain information such as job ID, required computing resources, and priority. Implement functions to enqueue new jobs, dequeue jobs for execution, and handle the circular nature of the queue. Additionally, provide functions to display the current job queue status and manage job dependencies.

Circular Resource Allocation:

You are building a resource allocation system for a cloud computing platform. Implement a circular linked list to manage the available resources. Each node in the linked list should represent a resource and contain information such as resource ID, type, and availability status. Develop functions to allocate resources to users, deallocate resources, and handle the circular nature of resource allocation. Additionally, provide functions to display the current resource allocation status and manage resource sharing between users.

QUEUES:

Supermarket Queue Management:

You have been hired to optimize the queue management system at a busy supermarket. Design a queue data structure to handle customers at the checkout counters. Implement functions to add customers to the queue, remove customers when they complete their checkout, and calculate the average waiting time for customers. Additionally, provide a priority queue implementation to prioritize certain types of customers, such as elderly or disabled individuals, to ensure they are served quickly.

Printer Spooling System:

You are tasked with developing a printer spooling system for a large office. Implement a queue data structure to manage print job requests from different users. Each print job contains information such as document name, number of pages, and priority level. Write functions to enqueue print jobs, dequeue jobs for printing, and handle job priorities. Additionally, provide a function to estimate the total printing time based on the job queue and printer speed.

Ticket Reservation System:

You are working on a ticket reservation system for a concert venue. Design a queue data structure to manage ticket requests from customers. Each ticket request includes information such as customer name, number of tickets, and seat preference. Implement functions to enqueue ticket requests, dequeue requests when tickets are available, and handle seat allocation based on customer preferences. Additionally, provide a function to display the current ticket queue status and optimize seat allocation for groups.

Traffic Signal Control:

You are developing a traffic signal control system for a busy intersection. Implement a queue data structure to manage the vehicles waiting at each lane. Each vehicle is represented by its ID and arrival time. Write functions to enqueue vehicles when they arrive, dequeue vehicles when the signal turns green, and maintain the order of vehicles based on their arrival time. Additionally, provide a function to estimate the average waiting time for vehicles at each lane.

Process Scheduling:

You are building an operating system's process scheduling algorithm. Design a queue data structure to manage the processes in the system. Each process is represented by its ID, arrival time, execution time, and priority. Implement functions to enqueue processes, dequeue processes for execution, and handle process priorities and execution order. Additionally, provide a function to calculate the average turnaround time for the processes.

Call Center Support:

You are developing a call center support system for a telecommunications company. Implement a queue data structure to manage incoming customer calls. Each call contains information such as caller ID, issue description, and priority level. Write functions to enqueue calls, dequeue calls for support agents, and prioritize urgent calls. Additionally, provide a function to calculate the average waiting time for customers.

Food Delivery Service:

You are building a food delivery service application. Design a queue data structure to manage incoming food orders from customers. Each order includes details like customer name, delivery address, and order items. Implement functions to enqueue orders, dequeue orders for delivery drivers, and optimize the delivery route based on order locations. Additionally, provide a function to estimate the average delivery time for orders.

Event Ticketing System:

You are working on an event ticketing system for a popular concert venue. Develop a queue data structure to manage ticket purchases from customers. Each ticket purchase includes information such as customer name, number of tickets, and payment details. Write functions to enqueue ticket purchases, dequeue purchases for processing, and handle payment verification. Additionally, provide a function to generate a waiting list for sold-out events.

Task Management Application:

You are building a task management application for a project team. Implement a queue data structure to manage tasks assigned to team members. Each task includes details such as task name, deadline, and assigned team member. Write functions to enqueue tasks, dequeue tasks for team members, and prioritize urgent tasks. Additionally, provide a function to calculate the average completion time for tasks.

Airport Security Check:

You are developing a security check system for an airport. Design a queue data structure to manage passengers going through the security screening process. Each passenger is represented by their boarding pass information, including name, flight details, and security status. Implement functions to enqueue passengers, dequeue passengers for screening, and handle security checks based on priority or random selection. Additionally, provide a function to estimate the average waiting time for passengers.

STACKS:

Elevator Dispatch System:

You are designing an elevator dispatch system for a high-rise building. Implement a stack data structure to manage the requests from different floors. Each floor request is represented as a node in the stack, containing information such as the floor number, direction (up or down), and timestamp. Implement functions to push new requests onto the stack, pop the most recent request when an elevator becomes available, and dispatch the elevator accordingly. Additionally, provide a function to display the current status of the elevator system.

Symbol Table in Compiler Design:

You are working on the symbol table component of a compiler. Use a stack data structure to implement a symbol table that stores information about variables, functions, and other symbols in the program. Each symbol is represented as a node in the stack, containing details such as the symbol name, data type, and scope. Implement functions to push new symbols onto the stack when encountered in the program, pop the stack when leaving a scope, and perform symbol lookup efficiently. Additionally, provide a function to display the current symbol table.

Undo/Redo in Graphic Design Software:

You are developing a graphic design software and want to provide undo/redo functionality for the user's actions. Use a stack data structure to store the history of design changes. Each design change is represented as a node in the stack, containing information such as the modified elements, properties, and timestamp. Implement functions to push new design changes onto the stack, pop the most recent change when undo is requested, and reapply a previously undone change when redo is triggered. Additionally, provide a function to display the current design state.

Train Management System:

You are tasked with designing a train management system for a busy railway station. Implement a stack data structure to manage the arrival and departure of trains. Each train is represented as a node in the stack, containing information such as the train number, arrival time, departure time, and platform number. Implement functions to push new trains onto the stack when they arrive, pop the stack when they depart, and display the current train schedule. Additionally, provide a function to search for a specific train by its number.

Sudoku Solver:

You are developing a Sudoku-solving algorithm using a stack data structure. The objective is to solve a partially filled Sudoku grid. Use a stack to keep track of the cells that need to be filled and the possible values that can be assigned to each cell. Implement functions to push cells onto the stack, pop the stack when backtracking is required, and fill the cells with valid values according to the Sudoku rules. Additionally, provide a function to display the solved Sudoku grid.

Valid Parentheses:

Write a program to check if a given string of parentheses is valid. The string can contain opening and closing parentheses, along with other characters. The program should determine if the parentheses are balanced and properly nested. Implement a stack data structure to solve this problem efficiently.

Reverse Polish Notation:

Design an algorithm to evaluate arithmetic expressions in Reverse Polish Notation (RPN). RPN is a mathematical notation in which every operator follows all of its operands. Implement a stack-based solution to parse and evaluate RPN expressions, supporting operators like addition, subtraction, multiplication, and division.

Next Greater Element:

Given an array of integers, find the next greater element for each element in the array. The next greater element is the first element to the right that is greater than the current element. Implement a stack-based solution to efficiently solve this problem, returning an array of the next greater elements for each input element.

Largest Rectangle in a Histogram:

Given a histogram represented by an array of heights, find the area of the largest rectangle that can be formed within the histogram. The rectangle must be aligned with the histogram's bars and can span multiple bars. Implement a stack-based solution to efficiently find the maximum area of the rectangle.

Design a Min Stack:

Design a stack that supports push, pop, and getMin operations in constant time. The getMin operation should return the minimum element in the stack. Implement the stack using an additional stack or modify the node structure to include an extra field for the minimum value.