**SINGLY-LINKED LIST**  
1. **E-commerce shopping cart:**  
You are creating a shopping cart system for an online store. The cart needs to manage items efficiently and calculate the total price. Implement a singly linked list to keep track of the items in the cart. Each item should have a name, price, and quantity. Write functions to add items to the cart, remove items by name, and calculate the total price of all items in the cart.

NOTE: In the given problem statement, a singly linked list is used as the data structure to keep track of the items in the shopping cart system for an online store. A singly linked list is a linear data structure where each element (node) contains data and a reference to the next node in the list. In this case, each node of the linked list represents an item in the shopping cart and contains attributes such as the item's name, price, and quantity. By utilizing a singly linked list as the data structure, the shopping cart system can effectively manage the items, provide functions for adding and removing items, and calculate the total price of all 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.

NOTE: In the given problem statement, a singly linked list is used as the data structure to store the lines of a document efficiently in a text editor application. Each node in the linked list represents a line of text, and the order of the lines is maintained. By using a singly linked list, the text editor application can efficiently handle large documents while providing the required operations such as insertion, deletion, searching, and counting of lines  
  
3. **Social Media Platform:**   
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 and follower count. Write functions to add followers, remove followers, count the number of followers for each user and display the followers of a particular user. Additionally, provide a function to find the common followers between two users.

NOTE: In the given problem statement, a singly linked list is used as the data structure to represent the followers of a user in a social media application.  
  
4. **Movie ticket booking at the theatre:**  
You are creating a ticket booking system for a theatre with many seats. Implement a list to manage the available seats. Each seat should have a number, row, and availability status. Write functions to reserve a seat, cancel a reservation, display the seating arrangement, and check the number of available seats

NOTE: By utilizing a Singly Linked list as the data structure, the ticket booking system can efficiently manage the available seats, handle reservations and cancellations, display the seating arrangement, and determine the number of available seats.  
  
5. **Delivery Service Navigation System:**   
You are developing a delivery service navigation system for a city. Implement a list to represent the stops on the delivery route. Each stop should have an address. Write functions to add stops to the route, remove stops, count the number of stops and check if a specific stop is on the route.

NOTE: In the given problem statement, a list is used as the data structure to represent the stops on the delivery route in the navigation system for a city. A Singly Linked list is a fundamental data structure in programming that allows for the storage and retrieval of elements in a specific order. In this case, the list is used to maintain the sequence of stops on the delivery route, with each element in the list representing a stop and containing the address information.

**6. AI Assistant:**  
You are creating a scheduling application for a busy professional. Use a list to store the appointments for a day. Each appointment will have a start time, end time, and description. Write functions to add appointments, delete appointments, check for any overlapping appointments, display the day's schedule in order, and find the earliest appointment.

NOTE: By using a Singly Linked list as the data structure, the scheduling application can effectively manage the appointments, provide functionality for adding and deleting appointments, check for overlaps, display the schedule in order, and find the earliest appointment.  
  
7. **University Students 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, total marks, and enrollment status. Develop functions to add students, remove students by their ID, find the total number of students, find the average marks of the batch and display the student with the highest grade. Take care of the situation where two students have the same score and that score is the highest mark.

NOTE: In the given problem statement, a singly linked list is used as the data structure to store the student records in the student management system for a university. A singly linked list is a linear data structure where each element (node) contains data and a reference to the next node in the list. In this case, each node of the linked list represents a student and contains attributes such as name, ID, total marks, and enrollment status. By utilizing a singly linked list as the data structure, the student management system can effectively store and manage the student records, provide functions for adding and removing students, calculate aggregate information such as the total number of students and average marks, and identify the student with the highest grade.

**8. DNA Sequence Analysis:**  
You are working on a project that involves analyzing DNA sequences. To accomplish this, you need to create a data structure using a list. Each element in the list should represent a nucleotide (A, T, C, or G) and its relationship to the next nucleotide. Write functions to combine two DNA sequences, search for specific patterns in the sequences, and calculate the percentage of GC content in a given sequence.

NOTE: In the given problem statement, a Singly linked list is used as the data structure to represent DNA sequences and their relationships. A list is a commonly used data structure that allows for the storage and retrieval of elements in a specific order. In this case, each element in the list represents a nucleotide (A, T, C, or G) and its relationship to the next nucleotide. By using a list as the data structure, the DNA analysis project can effectively store and analyze DNA sequences, perform operations such as combining sequences, searching for patterns, and calculating the GC content percentage.

**9. Cryptocurrency Transaction Ledger:**

You're developing a cryptocurrency transaction ledger and need to ensure its integrity. The ledger will be represented using a list, where each element contains transaction details like the sender's address, the receiver's address, and the amount transferred. Your task is to implement a record count verification system to validate the ledger's integrity. To perform record count verification, maintain a count of the total number of transactions in the ledger. Whenever a new transaction is added or removed, update the record count accordingly. At regular intervals, compare the current record count with the expected count. If they match, it indicates that the ledger remains intact without any unauthorized modifications. The record count verification system provides a simple way to ensure the integrity of the ledger. By comparing the record count before and after any changes, you can quickly identify if transactions have been added or removed. If the count matches, the ledger can be considered trustworthy. By implementing this system, you can maintain the integrity of the cryptocurrency transaction ledger and ensure that all transactions are accounted for.

NOTE: In the given problem statement, a list is used as the data structure to represent the cryptocurrency transaction ledger. A singly linked list is a commonly used data structure that allows for the storage and retrieval of elements in a specific order. In this case, each element in the list represents a transaction in the cryptocurrency ledger and contains details such as the sender's address, receiver's address, and the amount transferred. By utilizing a list as the data structure, the ledger can effectively store the transaction details and provide a mechanism to verify the integrity of the ledger by comparing the record count.

**10. Phone Contact Management:**

You are developing a phone contact management system. Implement a singly linked list to store the contact details. Each node in the linked list should represent a contact and contain information such as name, phone number, and email address. Write functions to add contacts, search for contacts by name, delete contacts by name, count the total number of contacts and display the list of contacts.

NOTE: In the given problem statement, a singly linked list is used as the data structure to store the contact details in the phone contact management system. A singly linked list is a linear data structure where each element (node) contains data and a reference to the next node in the list. In this case, each node of the linked list represents a contact and contains attributes such as name, phone number, and email address. By using a singly linked list as the data structure, the phone contact management system can effectively store and manage the contact details, provide functionality for adding, searching, and deleting contacts, count the total number of contacts, and display the list of contacts.

**CIRCULAR LINKED LIST**1. **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.

NOTE: In the given problem statement, a circular linked list is used as the data structure to manage the list of processes in the operating system scheduler using the round-robin scheduling algorithm. A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a process and contains information such as the process ID, execution time, and priority. The circular linked list is used to efficiently manage the list of processes in the scheduler, allowing for rotation of the execution order based on a fixed time quantum. The functions in the scheduler can be implemented using linked list operations. By utilizing a circular linked list as the data structure, the operating system scheduler can effectively manage the list of processes, implement round-robin scheduling, add and remove processes, rotate the execution order, display the scheduling order, and handle priority adjustments.

**2. 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.

NOTE: In the given problem statement, a circular linked list is used as the data structure to represent the songs in the playlist for the music player application with a circular playlist feature. A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a song and contains information such as the song name, artist, and duration. The circular linked list is used to efficiently manage the playlist of songs, allowing for seamless looping and continuous playback. By utilizing a circular linked list as the data structure, the music player application can effectively manage the playlist of songs, support seamless looping, add and remove songs, skip to the next song, and display the current playing song.

**3. 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.

NOTE: By utilizing a circular linked list as the data structure, the task scheduler can effectively manage the tasks, implement a circular execution order based on priorities, add and remove tasks, rotate the execution order, display the task schedule, and handle task dependencies in the embedded system.

**4. Circular Job:**

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.

NOTE: A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a job and contains information such as the job ID, required computing resources, and priority. By utilizing a circular linked list as the data structure, the job queue can effectively manage the jobs, support circularity in the queue, enqueue and dequeue jobs based on priorities, display the job queue status, and handle job dependencies in the distributed computing system.

**5. 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.

NOTE: A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a resource and contains information such as the resource ID, type, and availability status. The circular linked list is used to efficiently manage the available resources, allowing for the circular nature of resource allocation and ensuring fair distribution among users. By utilizing a circular linked list as the data structure, the resource allocation system can effectively manage the available resources, support circularity in resource allocation, allocate and deallocate resources, display the resource allocation status, and handle resource sharing between users in the cloud computing platform.   
  
**6. Restaurant Waitlist:**

You are developing a waitlist management system for a busy restaurant. Implement a circular linked list to manage the customers waiting for a table. Each node in the linked list should represent a customer and contain information such as their name, party size, and estimated wait time. Write functions to add customers to the waitlist, remove customers when their table is ready, and navigate through the waitlist in a circular manner. Additionally, provide functions to display the current waitlist status and notify customers when their table is ready.

NOTE: A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a customer and contains information such as their name, party size, and estimated wait time. The circular linked list is used to efficiently manage the waitlist, allowing for the circular navigation through the customers and ensuring fair and continuous management of the waitlist. By utilizing a circular linked list as the data structure, the waitlist management system can effectively manage the customers, support circular navigation through the waitlist, add and remove customers, display the waitlist status, and notify customers when their table is ready in the busy restaurant.  
  
**7. Movie Rental System:**

You are developing a movie rental system for a video rental store. Implement a circular linked list to manage the inventory of movies available for rent. Each node in the linked list represents a movie and contains information such as the movie title, genre, and availability status. Implement functions to add movies to the inventory, remove movies, and navigate through the movie collection in a circular manner. Additionally, provide functions to search for movies by title or genre, rent movies to customers, and display the currently available movies.

NOTE: A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a movie and contains information such as the movie title, genre, and availability status. The circular linked list is used to efficiently manage the movie inventory, allowing for circular navigation through the movies and ensuring continuous access to the collection. By utilizing a circular linked list as the data structure, the movie rental system can effectively manage the inventory of movies, support circular navigation through the movie collection, add and remove movies, search for movies by title or genre, rent movies to customers, and display the currently available movies in the video rental store.  
  
**8. Parking Lot Management System:**

You are tasked with designing a parking lot management system for a commercial complex. Implement a circular linked list to represent the available parking spaces. Each node in the linked list represents a parking space and contains information such as the space number, vehicle details, and availability status. Implement functions to add vehicles to the parking lot, remove vehicles when they leave, and find the nearest available parking space. Additionally, provide functions to display the current occupancy of the parking lot and optimize the parking space allocation for efficient utilization.

NOTE: A circular linked list is a variation of a singly linked list where the last node points back to the first node, creating a circular structure. In this case, each node in the circular linked list represents a parking space and contains information such as the space number, vehicle details, and availability status. The circular linked list is used to efficiently manage the parking lot spaces, allowing for circular navigation through the parking spaces and ensuring continuous access to the spaces. The functions in the parking lot management system can be implemented using linked list operations

**9. Theme Park:**

You are creating a ticketing system for a theme park using a circular queue. Design a circular linked list to manage the tickets in the queue. Each ticket contains information such as ticket ID, customer details, and ride preference. Write functions to add tickets to the queue, remove tickets for processing, and handle the circular nature of the queue. Also, provide functions to display the current queue status and manage priority ticketing if needed.

NOTE: By using a circular linked list-based circular queue as the data structure, the ticketing system can efficiently manage the tickets, handle the circular nature of the queue, add and remove tickets, display the current queue status, and implement priority ticketing if necessary

**10. Employee Shift Scheduling:**

You are developing a shift scheduling system for a company. Implement a circular linked list to represent the shifts for employees. Each node in the linked list should contain information about a shift, such as start time, end time, and the employee assigned to that shift. The linked list should wrap around, allowing the shifts to cycle indefinitely. Implement functions to add new shifts, remove shifts, and assign employees to shifts. Additionally, provide functions to retrieve the next shift for a specific employee and to display the entire shift schedule.

NOTE: By using a circular linked list as the data structure, the shift scheduling system can efficiently manage the shifts, allow for continuous cycling of the shifts, add and remove shifts, assign employees to shifts, retrieve the next shift for a specific employee, and display the entire shift schedule.

**DOUBLY LINKED LIST  
  
1. 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 list of books. Additionally, provide functions to search for books by title and update the book information.

NOTE: By using a doubly linked list as the data structure, the book management system can efficiently store and manage the collection of books in the library. The doubly linked list allows for easy insertion and removal of books, bidirectional traversal of the list, searching for books by title, displaying the list of books, and updating book information.  
  
**2. Image Editing Application:**

You are developing an image editing application that requires managing layers efficiently. Create a doubly linked list to represent the layers, where each layer contains information such as layer name and visibility. Implement functions to add new layers, remove layers, and display the list of layers. Additionally, provide functions to toggle the visibility of layers, rename layers, and duplicate layers.

NOTE: By using a doubly linked list as the data structure, the image editing application can efficiently manage layers. The doubly linked list allows for easy insertion and removal of layers, bidirectional traversal of the list, toggling visibility, renaming layers, duplicating layers, and displaying the list of layers.  
  
**3. 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, departure station, and destination 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.

NOTE: By using a doubly linked list as the data structure, the train reservation system can efficiently manage passenger reservations. The doubly linked list allows for easy insertion and removal of reservations, bidirectional traversal of the list, displaying passenger details in reservation order, and finding the number of available seats on a specific train.   
  
**4. 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.

NOTE: By using a doubly linked list as the data structure, the task-scheduling application can efficiently manage tasks. The doubly linked list allows for easy insertion and removal of tasks, bidirectional traversal of the list, reordering tasks based on priority and deadline, and displaying tasks in chronological order with the ability to highlight overdue tasks.  
  
**5. 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, departure square 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.

NOTE: By using a doubly linked list as the data structure, the online multiplayer chess game platform can efficiently manage game moves. The doubly linked list allows for easy addition and removal of moves, bidirectional traversal of the list for undoing moves or replaying the game, validating move legality based on the game rules, and displaying the board position after each move.

**6. 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.

NOTE: By using a doubly linked list as the data structure, the hotel room management system can efficiently manage the rooms and their occupancy status. The doubly linked list allows for easy addition and removal of rooms, bidirectional traversal of the list for searching and updating room details, marking rooms as occupied or vacant, checking guests in and out, and displaying the current room occupancy status.   
  
**7. 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.

NOTE: By using a doubly linked list as the data structure, the music player application can efficiently manage playlists, support navigation between playlists in both directions, add and remove songs from playlists, and provide seamless playback between songs in different playlists. The bidirectional traversal capability of the doubly linked list allows for easy playlist management and advanced playlist navigation features in the application.  
  
**8. 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, departure address 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 a function to handle exceptions like mishandled baggage.

NOTE: By using a doubly linked list as the data structure, the baggage handling system can efficiently manage the flow of baggage from check-in to the boarding gate. The bidirectional traversal capability of the doubly linked list allows for easy movement tracking, accurate delivery, and exception handling. The system can add baggage items, track their movement between checkpoints, ensure proper delivery to the appropriate boarding gate, and handle exceptions like mishandled baggage effectively.  
  
**9. 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 travelled by each vehicle and optimize route planning for efficient logistics operations.

NOTE: By using a doubly linked list as the data structure, the vehicle fleet management system can efficiently manage the fleet of vehicles for the logistics company. The bidirectional traversal capability of the doubly linked list allows for easy movement tracking, removal of vehicles, and efficient route planning. The system can add new vehicles, track their movement between locations, calculate the distance travelled, and optimize route planning to streamline logistics operations effectively.

**10. 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 answers. 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.

NOTE: By using a doubly linked list as the data structure, the online quiz platform can efficiently manage the quiz questions, allow users to navigate through the questions in both directions, and provide scoring, result display, and feedback functionalities. The bidirectional traversal capability of the doubly linked list allows for easy question navigation, deletion, and efficient performance evaluation. The platform can add new questions, delete questions, allow users to navigate through the quiz, score the quizzes, display the results, and provide feedback on the user's performance effectively.

**QUEUE DATA STRUCTURE**

**1. 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 counter. Implement functions to add customers to the queue(enqueue), remove customers(deque) 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.

NOTE: To optimize the queue management system, a priority queue data structure can be used. A priority queue assigns a priority value to each customer, allowing for prioritization based on specific criteria such as customer type or urgency. In this case, the priority queue can be utilized to prioritize elderly or disabled individuals, ensuring they are served quickly by placing them at the front of the queue. The priority queue implementation maintains the order of customers based on their priority and facilitates efficient retrieval of the customer with the highest priority. This enables the supermarket to provide better service to prioritized customers while maintaining a fair and efficient queue management system  
  
**2. 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.

NOTE: A queue data structure is used to manage print job requests in a printer spooling system. Print jobs are enqueued and dequeued based on the FIFO principle, ensuring the order of processing. Job priorities can be handled by assigning priority levels, allowing higher priority jobs to be processed first. Additionally, the queue facilitates estimating the total printing time by considering the number of pages and printer speed.

**3. 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.

NOTE: A queue data structure is used to manage ticket requests in a ticket reservation system for a concert venue. Ticket requests are enqueued and dequeued based on the FIFO principle, ensuring a fair order of processing. The queue facilitates handling seat allocation by considering customer preferences such as seat preference. Functions are implemented to enqueue ticket requests, dequeue them when tickets are available, and display the current status of the ticket queue.  
  
**4. 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 in each lane.

NOTE: A queue data structure is used to manage vehicles at each lane in a traffic signal control system. Vehicles are enqueued as they arrive and dequeued when the signal turns green. The queue maintains the order of vehicles based on their arrival time, ensuring fairness in vehicle processing. Additionally, a function is provided to estimate the average waiting time for vehicles in each lane, helping to monitor and optimize traffic flow at the intersection.  
  
**5. 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 orders. Additionally, provide a function to calculate the average turnaround time (Turnaround time = completion time - arrival time)  for the processes.

NOTE: A queue data structure is used to manage processes in an operating system's process scheduling algorithm. Processes are enqueued based on their arrival time and dequeued for execution. The queue allows for handling process priorities and execution orders, ensuring fairness in process scheduling. Additionally, a function is provided to calculate the average turnaround time for the processes, which helps evaluate the efficiency of the scheduling algorithm by measuring the time taken to complete each process.  
  
**6. 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.

NOTE: A queue data structure is used to manage incoming customer calls in a call center support system. Calls are enqueued based on their arrival time and dequeued for support agents to handle. The queue allows for prioritizing urgent calls, ensuring that they are addressed promptly. Additionally, a function is provided to calculate the average waiting time for customers, which helps monitor and optimize the call center's performance in terms of response time and customer satisfaction.  
  
**7. Food Delivery Service for a Restaurant:**

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 ordered 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.

NOTE: A queue data structure is used to manage incoming food orders in a food delivery service application. Orders are enqueued as they arrive and dequeued for delivery drivers to handle. The queue allows for a fair and organized distribution of orders among the drivers. Additionally, the application can optimize the delivery route based on the order locations, ensuring efficient and timely deliveries. A function is provided to estimate the average delivery time for orders, which helps set realistic expectations for customers and evaluate the performance of the delivery service.  
  
**8. 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.

NOTE: A queue data structure is used to manage ticket purchases in an event ticketing system. Ticket purchases are enqueued as customers make their purchases, and they are dequeued for processing, allowing the system to handle them in the order they were received. The queue also facilitates payment verification for each purchase, ensuring the validity of the transactions. Additionally, a waiting list can be generated for sold-out events, where customers can be added to the queue in case tickets become available due to cancellations or additional releases.  
 **9. 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.

NOTE: A queue data structure is used to manage tasks in a task management application. Tasks are enqueued as they are assigned to team members, and they are dequeued for team members to work on them in the order they were received. The queue allows for prioritization of urgent tasks, ensuring that they are addressed promptly. Additionally, the average completion time for tasks can be calculated by tracking the time taken to complete each task and dividing it by the total number of completed tasks. This provides insights into the efficiency and productivity of the project team.

10.  **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.

NOTE: A queue data structure is used to manage the flow of passengers in a security check system at an airport. Passengers are enqueued as they arrive for screening, and they are dequeued to go through the security checks in the order they arrived. The queue allows for efficient and fair handling of passengers. The security checks can be performed based on priority, such as giving priority to passengers with special needs or those on tight flight schedules, or through random selection to ensure thorough screening. Additionally, the average waiting time for passengers can be estimated by tracking the time each passenger spends in the queue and dividing it by the total number of passengers screened. This helps in assessing the efficiency of the security check process.

**STACKS**  
  
**1. 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.

NOTE: A stack data structure is used to manage floor requests in an elevator dispatch system for a high-rise building. Each floor request is pushed onto the stack, containing information such as the floor number, direction (up or down), and timestamp. When an elevator becomes available, the most recent request is popped from the stack and the elevator is dispatched accordingly. The stack follows the Last-In-First-Out (LIFO) principle, ensuring that the most recent request is processed first. Additionally, a function can be implemented to display the current status of the elevator, providing information such as its current floor and direction of travel.

**2. 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.

NOTE: A stack data structure can be used to implement the symbol table component of a compiler. Each symbol encountered in the program is represented as a node in the stack, containing information such as the symbol name, data type, and scope. When encountering a new symbol, it is pushed onto the stack. When leaving a scope, symbols belonging to that scope are popped from the stack. This allows for efficient management of symbol scopes. Symbol lookup can be performed by traversing the stack from top to bottom, searching for the desired symbol based on its name and scope. Additionally, a function can be implemented to display the current symbol table, providing a visual representation of the stored symbols.  
  
**3. Undo/Redo in Graphic Design Software**:  
You are developing 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 timestamps. 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.

NOTE: To provide undo/redo functionality in a graphic design software, a stack data structure can be used 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 timestamps. When a new design change is made, it is pushed onto the stack. When the user requests an undo operation, the most recent change is popped from the stack, reverting the design to its previous state. Similarly, when the user requests a redo operation, a previously undone change is reapplied by pushing it back onto the stack. This allows for easy navigation through the design history. Additionally, a function can be implemented to display the current state of the design, providing the user with a visual representation of the design at any given point in time.

**4. Train Schedule Management:**  
You are creating a train schedule management system for a busy railway station. Use a stack to handle the arrival and departure of trains. Each train is represented as a node in the stack, with information like train number, arrival time, departure time, and platform number. Create functions to add new trains when they arrive, remove trains when they depart, and display the current train schedule. Also, include a function to search for a specific train by its number.  
  
NOTE: In a train schedule management system, a stack data structure can be used to handle the arrival and departure of trains. Each train is represented as a node in the stack, storing information such as the train number, arrival time, departure time, and platform number. When a train arrives, it is added to the stack using the push operation. When a train departs, the most recently arrived train is removed from the stack using the pop operation. This ensures that the trains are managed in a Last-In-First-Out (LIFO) manner. Additionally, a function can be implemented to display the current train schedule, providing information about the trains in the stack. Furthermore, a search function can be included to search for a specific train by its number, allowing users to quickly locate a particular train in the schedule.  
  
**5. Browser History:**  
You are building a web browser and need to implement the back and forward navigation functionality. Use a stack data structure to store the URLs visited by the user. Each URL is represented as a node in the stack, containing information such as the website address and timestamp. Implement functions to push new URLs onto the stack when the user visits a page, pop the stack to navigate back to the previous page, and move forward to a previously visited page. Additionally, provide a function to display the current website URL.

NOTE: To implement the back and forward navigation functionality in a web browser, a stack data structure can be used to store the URLs visited by the user. Each URL is represented as a node in the stack, containing information such as the website address and timestamp. When the user visits a new page, the URL is pushed onto the stack using the push operation. To navigate back to the previous page, the most recently visited URL is popped from the stack using the pop operation. Similarly, to move forward to a previously visited page, the URLs that were popped can be pushed back onto the stack. This allows the browser to maintain a history of visited pages and enables seamless navigation. Additionally, a function can be implemented to display the current website URL, providing the user with the information about the currently visited page.

**6. 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.

NOTE: In the given statement, a stack data structure is used to solve the problem efficiently. The stack is used to store the opening parentheses encountered in the string. As we iterate through the string, if we encounter a closing parenthesis, we check if the top of the stack contains the corresponding opening parenthesis. If the stack is empty or the top of the stack doesn't match the closing parenthesis, the string is considered invalid. By using a stack, we can efficiently determine if the parentheses are balanced and properly nested in the given string.  
  
**7. 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.

NOTE: In the given statement, a stack is used to implement a special type of stack that supports push, pop, and getMin operations in constant time. To achieve this, an additional stack or modification to the node structure is used. The stack is modified to include an extra field that keeps track of the minimum value at each step. When pushing elements onto the stack, the minimum value is updated if the new element is smaller. The getMin operation simply returns the minimum value stored in the stack's top element.

**8. Largest rectangle in a histogram:**  
Find the largest rectangle that can be formed within a histogram, where the histogram is represented by an array of heights. The rectangle must be aligned with the bars of the histogram and can span multiple bars. Use a stack to efficiently solve this problem and determine the maximum area of the rectangle that can be formed within the histogram.

NOTE: In this problem, a stack data structure is used to efficiently find the largest rectangle that can be formed within a histogram. The stack helps in comparing the heights of the bars in the histogram array and calculating the maximum area of the rectangle. By storing the indices of the bars in non-decreasing order of their heights, the stack allows for an optimal solution to this problem.

**9. Programming language code:**

You are creating a system to validate the syntax of a programming language's code using a stack. Design a stack-based data structure that checks if all opening and closing brackets, parentheses, and braces in the code are properly balanced and nested. Implement functions to push opening symbols onto the stack and pop them when matching closing symbols are encountered. If the stack is empty at the end, the code is valid; otherwise, it contains syntax errors.

NOTE: A stack data structure is used to validate the syntax of a programming language's code. Opening symbols are pushed onto the stack, and closing symbols are matched and popped from the stack. If the stack is empty at the end, the code is valid; otherwise, it contains syntax errors. The stack enables efficient tracking and matching of symbols, ensuring proper code syntax.

**Maze solving algorithm:**You are building a maze-solving algorithm using a stack to keep track of the current path. Design a stack-based data structure to efficiently explore and backtrack through the maze. Implement functions to push the current position onto the stack, pop the stack to backtrack to the previous position, and update the current position based on the available directions. By using a stack, you can effectively explore different paths until the exit of the maze is found.

NOTE: A stack data structure is used to build a maze-solving algorithm. The stack keeps track of the current path by pushing the current position onto the stack and popping it to backtrack to the previous position. The current position is updated based on available directions to explore different paths. By using a stack, the algorithm efficiently explores and backtracks through the maze until the exit is found.