Ejercicio: Pila de libros

Exercise: Managing a Stack

Create a program that simulates a stack using the **push** and **pop** methods. The stack should store a collection of books. Users can perform the following actions:

1. Add a new book to the top of the stack.
2. Remove the book from the top of the stack.
3. Display the current stack of books.

Implement a loop that allows users to interact with the stack until they choose to exit.

Ejercicio: Pila de libros

Crea un programa que simule una pila usando los métodos **push** y **pop**. La pila debe contener una colección de libros. Los usuarios deben poder ejecutar las siguientes acciones:

1. Agregar un libro encima de la pila.
2. Quitar un libro de encima de la pila.
3. Mostrar la pila actual de libros.

Implementa un ciclo que permita a los usuarios interactuar con la pila hasta que decidan salir.

Ejercicio: Gestionando una lista de canciones

Exercise: Managing a Playlist

Imagine you are developing a music application, and you need to create a function to manage a playlist. Your function should take a playlist array, remove the first song using the `shift` method, add a new song to the beginning using the `unshift` method, and return the updated playlist.

Instructions:

1. Define a function named `managePlaylist` that takes two parameters: `playlist` (an array) and `newSong` (a string representing the new song to be added).
2. Inside the function, use the `shift` method to remove the first song from the playlist.
3. Use the `unshift` method to add the new song to the beginning of the playlist.
4. Return the updated playlist from the function.
5. In the example usage section, create an initial playlist array and a new song to add.
6. Call the `managePlaylist` function with the initial playlist and the new song.
7. Display the initial playlist, the new song to add, and the updated playlist.

Ejercicio: Gestionando una lista de canciones

Imagina que estás desarrollando una aplicación y necesitas crear una función para gestionar una lista de canciones. Tu función debería tomar un array de listas de reproducción, eliminar la primera canción usando el método shift, añadir una nueva canción al principio usando el método unshift, y devolver la lista de reproducción actualizada.

Instrucciones:

1. Define una función llamada `managePlaylist` que toma dos parámetros: `playlist` (un array) y `newSong` (una cadena que representa la nueva canción a añadir).
2. Dentro de la función, utiliza el método `shift` para eliminar la primera canción de la lista de reproducción.
3. Utilice el método `unshift` para añadir la nueva canción al principio de la lista de reproducción.
4. Devuelve la lista de reproducción actualizada desde la función.
5. En la sección de ejemplo de uso, crea una lista de reproducción inicial y una nueva canción para añadir.
6. Llama a la función `managePlaylist` con la lista de reproducción inicial y la nueva canción.
7. Muestra la lista de reproducción inicial, la nueva canción a añadir y la lista de reproducción actualizada.

Ejercicio: Implementación de un Juego de Cartas

Exercise: Card Game Implementation

Imagine you're building a simple card game. You have an array representing a deck of cards, and you want to perform the following operations:

1. Shuffle the Deck: Randomly rearrange the order of the cards in the deck.
2. Deal Cards: Deal a specific number of cards to players from the top of the deck. This exercise involves using the **splice()** method to shuffle the deck and deal cards.

Ejercicio: Implementación de un Juego de Cartas

Supongamos que estás construyendo un juego de cartas simple. Tienes un array que representa un mazo de cartas y deseas realizar las siguientes operaciones:

1. Barajar el Mazo: Reorganizar aleatoriamente el orden de las cartas en el mazo.
2. Repartir Cartas: Repartir una cantidad específica de cartas a los jugadores desde la parte superior del mazo. Este ejercicio implica el uso del método **splice()** para barajar el mazo y repartir las cartas.

Ejercicio: suma de Elementos en un Array

Sum of Elements in an Array

Create a program that takes an array of numbers as input and calculates the sum of all elements in that array.

Suma de Elementos en un Array

Crea un programa que tome un array de números como entrada y calcule la suma de todos los elementos en ese array.

Ejercicio: Calificación Promedio Aprobatoria

Passing Grade Average

Create a program that takes an array of grades as input and calculates the average only for passing grades (greater than or equal to 70).

Calificación Promedio Aprobatoria

Crea un programa que tome un array de calificaciones como entrada y calcule el promedio únicamente de las calificaciones aprobatorias (mayores o iguales a 70).

Ejercicio: Ganadores de Rifa

Raffle Winner Verification

Program In this program, you can verify if a person is among the list of winners in a raffle. Simply input the name or ticket number, and the program will check and display the winner's information.

Programa de Verificación de Ganadores de Rifa

En este programa, puedes verificar si una persona está entre los ganadores de una rifa. Simplemente ingresa el nombre o número de boleto, y el programa verificará y mostrará la información del ganador.

Ejercicio: Análisis de transacciones

Transaction analysis

Imagine you have a list of financial transactions and you want to perform various data processing operations. Use the following instructions as a guide to complete the exercise:

1. Calculate Total Balance: - Use the **reduce** method to calculate and display the total balance of all transactions.
2. Find the Largest Transaction (Income or Expense): - Utilize the **reduce** method to find the transaction with the largest amount (either income or expense) and display it in the console.
3. Filter Purchase Transactions: - Use the **filter** method to obtain and display in the console only the purchase transactions (with negative amounts).
4. Find a Transaction by Description: - Use the **find** method to search and display in the console a specific transaction by its description.
5. Find the Index of a Transaction by Amount: - Employ the **findIndex** method to find and display in the console the index of a specific transaction by its amount.
6. Update Transaction Descriptions: - Use the **forEach** method to update the descriptions of transactions. Add the prefix "Expense: " to transactions with negative amounts and "Income: " to transactions with positive amounts. Display the updated transactions in the console.

Remember to adapt and combine these operations as needed.

Análisis de transacciones

1. Imagina que tienes una lista de transacciones financieras y deseas realizar varias operaciones de procesamiento de datos. Usa las siguientes instrucciones como guía para completar el ejercicio:
2. Calcular el Saldo Total: - Utiliza el método **reduce** para calcular y mostrar el saldo total de todas las transacciones.
3. Encontrar la Transacción más Grande (Ingreso o Egreso): - Emplea el método **reduce** para encontrar la transacción con el mayor monto (ya sea ingreso o egreso) y muéstrala en la consola.
4. Filtrar Transacciones de Compra: - Usa el método **filter** para obtener y mostrar en la consola solo las transacciones de compra (con montos negativos).
5. Encontrar una Transacción por Descripción: - Utiliza el método **find** para buscar y mostrar en la consola una transacción específica por su descripción.
6. Encontrar el Índice de una Transacción por Monto: - Emplea el método **findIndex** para encontrar y mostrar en la consola el índice de una transacción específica por su monto.
7. Actualizar Descripciones de Transacciones: - Utiliza el método **forEach** para actualizar las descripciones de las transacciones. Añade el prefijo "Gasto: " a las transacciones con montos negativos y "Ingreso: " a las transacciones con montos positivos. Muestra las transacciones actualizadas en la consola.

Recuerda adaptar y combinar estas operaciones según sea necesario.

Ejercicio: Encontrar Índices de Subcadena

Finding Substring Indices

Given an array of strings and a target string, write a function to determine if the target string is present in the array. If it is present, return the index of the first occurrence and the index of the last occurrence; otherwise, return -1.

Encontrar Índices de Subcadena

Dado un array de cadenas de texto y una cadena de texto objetivo, escribe una función para determinar si la cadena objetivo está presente en el array. Si está presente, devuelve el índice de la primera ocurrencia y el índice de la última ocurrencia; de lo contrario, devuelve -1.

Ejercicio: Ganador del torneo

Algorithmic Tournament winner

An algorithmic tournament is underway, where teams of programmers compete to solve algorithmic problems as quickly as possible.

The competition follows a round-robin format, with each team facing off against all other teams. Only two teams compete against each other at a time, and in each competition, one team is designated as the home team, while the other is the away team. There is always a clear winner and loser in each competition, with no ties. Teams earn 3 points for a win and 0 points for a loss. The overall winner of the tournament is the team with the highest total points.

Your task is to write a function that determines the winner of the tournament based on the results of the competitions. The input consists of two arrays: **competitions** and **results**. The competitions array contains pairs of teams represented as **[homeTeam, awayTeam]**, where each team is a string of at most 30 characters. The **results** array indicates the winner of each corresponding competition in the competitions array. Specifically, **results[i]** denotes the winner of **competitions[i]**, where a 1 in the results array means that the home team won, and a 0 means that the away team won.

It is guaranteed that exactly one team will win the tournament, and each team will compete against all other teams exactly once. Additionally, the tournament will always have at least two teams.

Ganador del torneo algorítmico

Está teniendo lugar un torneo algorítmico en el que equipos de programadores compiten entre sí para resolver problemas algorítmicos lo más rápido posible. Los equipos compiten en un formato de todos contra todos, donde cada equipo se enfrenta a todos los demás. Solo dos equipos compiten entre sí en cada enfrentamiento, y en cada competición, un equipo es designado como equipo local, mientras que el otro es el equipo visitante. Siempre hay un claro ganador y perdedor en cada competición, sin empates. Los equipos obtienen 3 puntos por una victoria y 0 puntos por una derrota. El ganador general del torneo es el equipo con la mayor cantidad de puntos.

Tu tarea es escribir una función que determine al ganador del torneo en función de los resultados de las competiciones. La entrada consta de dos arrays: **competitions** y **results**. El array **competitions** contiene pares de equipos representados como **[equipoLocal, equipoVisitante]**, donde cada equipo es una cadena de hasta 30 caracteres. El array **results** indica el ganador de cada competición correspondiente en el array competitions. Específicamente, **results[i]** denota el ganador de **competitions[i]**, donde un 1 en el array results significa que el equipo local ganó y un 0 significa que el equipo visitante ganó.

Se garantiza que exactamente un equipo ganará el torneo, y cada equipo competirá contra todos los demás equipos exactamente una vez. Además, se garantiza que el torneo siempre tendrá al menos dos equipos.