Q1) Palindrome Checker :

Build a program that checks whether a given word or phrase is a palindrome. The user inputs a word or phrase, and the program determines if it reads the same forward and backwards, ignoring spaces and punctuation.

Code :

```
function isPalindrome(input) {
  // Remove spaces and punctuation, and convert to lowercase
  const cleanedInput = input.replace(/[^a-zA-Z0-9]/g, '').toLowerCase();
 // Compare the cleaned input with its reverse
  const reversedInput = cleanedInput.split('').reverse().join('');
  return cleanedInput === reversedInput;
}
// Test the function
const userInput = prompt("Enter a word or phrase:");
if (isPalindrome(userInput)) {
  console.log(`"${userInput}" is a palindrome!`);
} else {
  console.log(`"${userInput}" is not a palindrome.`);
}
```

Q2) Anagram Checker:

Write a function `isAnagram` which takes 2 parameters and returns true/false if those are anagrams or not.
What's Anagram?
- A word, phrase, or name formed by rearranging the letters of another, such as spar, formed from rasp.

Code :

```
function isAnagram(str1, str2) {
  // Remove spaces and punctuation, and convert to lowercase
  const cleanedStr1 = str1.replace(/[^a-zA-Z0-9]/g, '').toLowerCase();
  const cleanedStr2 = str2.replace(/[^a-zA-Z0-9]/g, '').toLowerCase();

  // Check if the lengths of the cleaned strings are equal
  if (cleanedStr1.length!== cleanedStr2.length) {
```

```
    return false;
  }
// Sort the characters in each string and compare
  const sortedStr1 = cleanedStr1.split('').sort().join('');
  const sortedStr2 = cleanedStr2.split('').sort().join('');
  return sortedStr1 === sortedStr2;
}
// Test the function
const str1 = "listen";
const str2 = "silent";
console.log(isAnagram(str1, str2)); // true
const str3 = "hello";
const str4 = "world";
console.log(isAnagram(str3, str4)); // false
```

Q3) Todo Implement using Class:

Implement a class `Todo` having the below methods
- **add(todo): adds todo to list of todos**
- **remove(indexOfTodo): remove todo from the list of todos**
- **update(index, updatedTodo): update todo at given index**
- **getAll: returns all todos**
- **get(indexOfTodo): returns todo at the given index**
- **clear: deletes all todos**

Code :

```
class Todo {
  constructor() {
    this.todos = [];
  }
add(todo) {
    this.todos.push(todo);
  }
remove(index) {
if (index >= 0 && index < this.todos.length) {
     this.todos.splice(index, 1);
```

```
    }
  }
update(index, updatedTodo) {
    if (index >= 0 && index < this.todos.length) {
      this.todos[index] = updatedTodo;
    }
  }
 getAll() {
    return [...this.todos]; // return a copy of the array
  }
 get(index) {
    if (index >= 0 && index < this.todos.length) {
      return this.todos[index];
    }
    return null;
  }

  clear() {
    this.todos = [];
  }
}
// Example usage:
const todoList = new Todo();

todoList.add("Buy milk");
todoList.add("Walk the dog");
todoList.add("Do laundry");
console.log(todoList.getAll()); // ["Buy milk", "Walk the dog", "Do laundry"]
todoList.update(1, "Walk the cat");
console.log(todoList.getAll()); // ["Buy milk", "Walk the cat", "Do laundry"]
```

```
todoList.remove(0);
console.log(todoList.getAll()); // ["Walk the cat", "Do laundry"]
console.log(todoList.get(0)); // "Walk the cat"
todoList.clear();
console.log(todoList.getAll()); // []
```