SET A

1. **Perform CRUD operation in JavaScript using Node.js** – implement Create, Read, Update, and Delete operations for patients' medical records.
2. Demonstrate an example for **import/export** in JavaScript (and TypeScript) for:
   a. Variable
   b. Method
   c. Class
   d. Provide command for installing TypeScript and running TypeScript code
3. Create a **Ward** class with a parameterized constructor to initialize wardName, capacity, and inChargeDoctor.
4. Define an abstract class **HospitalStaff** with abstract methods getSalary() and getRole(). Create derived classes **Surgeon** and **Receptionist** that implement these methods.
5. Create an interface **AppointmentManagement** with a method scheduleAppointment(). Implement this interface in the **Receptionist** class.
6. Create a generic function **getPatientInfo(info: T): T** that returns any type of patient information, such as name, ID, or diagnosis.
7. Create a Ward class with a parameterized constructor to initialize wardName, capacity, and inChargeDoctor.
8. Define an abstract class HospitalStaff with abstract methods getSalary() and getRole(). Create derived classes Surgeon and Receptionist that implement these methods.
9. Create an interface AppointmentManagement with a method scheduleAppointment(). Implement this interface in the Receptionist class.
10. Create a generic function getPatientInfo(info: T): T that returns any type of patient information, such as name, ID, or diagnosis.
11. Define a class named **Nurse** with attributes: `name`, `nurseId`, `department`, and `experience`. Include a method `displayInfo()`.
12. Write JavaScript to create a new <li> element representing a patient prescription, set its text content, and append it to an existing <ul>. Store the prescription in local storage.
13. Implement an event listener that updates and displays the total number of admitted patients when a button with ID "calculateTotalPatients" is clicked. Track the value in local storage.
14. Create a patient registration form with fields for name, email, and password. Implement validation for:
    a. Name: Should not be empty and contain only alphabetic characters.
    b. Email: Should be a valid email format.
    c. Password: Should be at least 8 characters long.
    d. Store patient data in local storage after validation.
15. Create a login form that checks patient credentials stored in local storage and displays an alert if incorrect.
16. Write JavaScript to display a counter for the number of appointments booked, allowing increment, decrement, and reset operations.
17. Use try/catch to handle errors during patient registration, such as missing insurance details or invalid contact info.
18. Demonstrate exception propagation when processing an emergency case where no ambulance is available.

19. Show an example of a finally block that executes when a patient is discharged, ensuring medical records are updated.

20. Create a user-defined exception called InvalidInsurancePolicy and demonstrate how it is used when a patient enters an expired insurance number.
21. Show an example of throw where a function throws an exception if a patient tries to book surgery outside hospital hours.

22. Create an arrow function named **calculateHealthRisk** that takes a BMI value as a parameter and returns the risk level (e.g., High, Medium, Low).
23. Implement an anonymous function assigned to the variable **increaseMedicineStock** that takes an array of medicine stock values and increases each by 5.
24. Write a named function **filterCriticalPatients** that takes an array of patient health scores and returns only those who need urgent care.
25. Demonstrate array methods (`map`, `filter`, `reduce`):
    a. Use `map` to convert patient temperatures to Fahrenheit.
    b. Use `filter` to extract patients with blood pressure above normal.
    c. Use `reduce` to calculate the total number of hospital visits.
26. Create a function **findMostVisitedDoctor** that accepts an array of visit counts and returns the doctor with maximum visits.
27. Demonstrate an example of array and object destructuring to extract **doctor names** from an array and their specialization from an object.
28. Demonstrate an example of a **Promise** that simulates checking appointment confirmation status for a patient.
29. Demonstrate an example of **async/await** to simulate fetching and displaying medical test results.
30. Show an example of **default and rest parameters** in a function that calculates the total treatment cost with additional charges.