Enunciado — Proyecto Node.js (TypeScript/JavaScript) con pruebas y tablas de prueba

1) Objetivo

Crear un proyecto en Node.js (TypeScript o JavaScript) que contenga:

- Una función esMayorDeEdad para determinar si una persona es mayor de edad.
- Una función puedeAprobarPrestamo que decida si se aprueba un préstamo, exclusivamente con base en las condiciones indicadas.
- Tablas de prueba: clases de equivalencia para esMayorDeEdad y tabla de decisiones para puedeAprobarPrestamo.
- Pruebas unitarias en Jest con cobertura del 100% de las funciones creadas.
- Video explicando el proyecto y las pruebas, con el estudiante visible en PiP durante todo el video.

2) Requisitos funcionales

2.1 Función 1: esMayorDeEdad

Firma sugerida (TypeScript):

```
export function esMayorDeEdad(edad: number): boolean;
```

Comportamiento:

- Retorna true si edad >= 18.
- Retorna false si 0 <= edad < 18.
- Entrada inválida (números negativos, NaN, null, undefined, no numérico): manejar de forma explícita (lanzar error o retornar un valor especial) y documentarlo.

2.2 Función 2: puedeAprobarPrestamo

Firma sugerida (TypeScript):

```
export type Ocupacion = 'estudiante' | 'empleado' | 'retirado';
export interface Persona {
  edad: number;
  ocupacion: Ocupacion;
  tieneDeudas: boolean; // deudas sin pagar
```

```
}
export function puedeAprobarPrestamo(persona: Persona): boolean;
```

Reglas (impleméntalas tal cual):

1. Si la persona es estudiante → NO se aprueba, aunque no tenga deudas.
2. Si la persona es empleado/trabajador → SÍ se aprueba, aunque tenga deudas.
3. Si la persona es retirado → SÍ se aprueba si no presenta deudas.
4. Si la persona es retirado → NO se aprueba si presenta deudas.

3) Requisitos técnicos del proyecto

- Node.js 18+ recomendado.
- Jest configurado (ts-jest si usas TypeScript).
- Cobertura al 100% en líneas, funciones, ramas y sentencias para los archivos de las funciones.

Estructura mínima sugerida:

```
/src
  /domain
    mayor-edad.ts|js
    prestamo.ts|js
    types.ts|js
  /tests
    mayor-edad.spec.ts|js
    prestamo.spec.ts|js
jest.config.ts|js
package.json
README.md
/docs
  equivalencias-mayor-edad.md
  decision-table-prestamo.md
```

Ejemplo de configuración de cobertura en package.json:

```
"jest": {
  "collectCoverage": true,
  "collectCoverageFrom": ["src/domain/**/*.*"],
  "coverageThreshold": {
    "global": { "branches": 100, "functions": 100, "lines": 100,
"statements": 100 }
  }
}
```

Scripts sugeridos:

```
"scripts": {
  "test": "jest",
  "test:cov": "jest --coverage"
}
```

4) Pruebas unitarias (Jest)

- Probar todas las clases de equivalencia y reglas de la tabla de decisiones.
- Incluir pruebas de frontera (17 y 18) y de errores (edad negativa, NaN, etc.).
- Alcanzar 100% de cobertura para los archivos de las funciones.
- Nombrar tests de forma clara (ej.: "empleado con deudas debe aprobarse").

5) Entregables

- Repositorio con código fuente, configuración de Jest, package.json y README.md.
- Carpeta /docs con las tablas (Markdown, PDF o imagen legible).
- Video con PiP (YouTube “No listado” o Drive/OneDrive “cualquier persona con el enlace: lector”) mostrando cámara todo el tiempo, explicación de funciones y tablas, y ejecución de las pruebas con el reporte de cobertura. Duración sugerida: 4–8 minutos.
- **PDF que contenga la URL del repositorio y URL del video.**