**1. Alcance y Tipo de Problemas a Diagnosticar:**

* **Empieza por lo básico:** No intentes cubrir todos los problemas posibles al inicio. Enfócate en un conjunto limitado de fallas comunes y bien definidas.
  + **Hardware:**
    - La computadora no enciende.
    - No hay señal de video.
    - No reconoce el disco duro/SSD.
    - Problemas de conectividad Wi-Fi/Ethernet.
    - La computadora se apaga inesperadamente.
    - Sonidos inusuales (bips del BIOS, ventiladores ruidosos).
  + **Software:**
    - Sistema operativo no carga.
    - Aplicaciones que no responden o se cierran solas.
    - Problemas de navegación web (lenta, no carga páginas).
    - Mensajes de error específicos del sistema.
    - Rendimiento lento general del sistema.
    - Infecciones de malware (aunque esto puede ser más complejo).
* **Define la profundidad del diagnóstico:** ¿El sistema solo identificará el componente o intentará dar pasos para la solución? Al principio, concéntrate en la identificación.

**2. Componentes Clave de un Sistema Experto (y cómo aplicarlos):**

* **Base de Conocimiento (Knowledge Base):** Aquí es donde residen las "reglas básicas" que mencionas. Puedes representarlas de varias formas:
  + **Reglas de Producción (IF-THEN):** Las más comunes para este tipo de sistema.
    - SI "La computadora no enciende" Y "La fuente de poder no está conectada" ENTONCES "Conectar la fuente de poder"
    - SI "No hay señal de video" Y "El monitor está apagado" ENTONCES "Encender el monitor"
    - SI "La computadora hace bips al encender" Y "Los bips son 3 cortos" ENTONCES "Problema de memoria RAM"
    - SI "Internet no funciona" Y "Otros dispositivos sí tienen internet" Y "No hay IP asignada" ENTONCES "Revisar configuración de red en la computadora"
  + **Hechos (Facts):** La información que el usuario proporciona al sistema.
    - "La pantalla está en negro."
    - "Se escuchan ruidos extraños."
    - "Puedo navegar en mi teléfono pero no en la computadora."
* **Motor de Inferencia (Inference Engine):** Es el "cerebro" que usa la base de conocimiento para llegar a conclusiones.
  + **Encadenamiento hacia adelante (Forward Chaining):** Empieza con los hechos (lo que el usuario observa) y deduce nuevas conclusiones hasta llegar a un diagnóstico. Ejemplo: "El usuario dice que la PC no enciende". El motor busca reglas que comienzan con "Si la PC no enciende..." y pide más información al usuario si es necesario.
  + **Encadenamiento hacia atrás (Backward Chaining):** Empieza con un posible diagnóstico y busca las reglas que lo soportarían, preguntando al usuario por los hechos necesarios. Ejemplo: "¿Es un problema de RAM?". El motor pregunta: "¿Hace la PC bips al encender?" y si es así, "¿Cuántos bips?".
* **Interfaz de Usuario (User Interface):** Cómo interactúa el usuario con el sistema.
  + Preguntas claras y concisas al usuario.
  + Opciones de respuesta (sí/no, selección múltiple).
  + Presentación del diagnóstico y posibles soluciones/siguientes pasos.
* **Módulo de Adquisición de Conocimiento (Knowledge Acquisition Module):** (Opcional para un proyecto inicial) Cómo se agregan, modifican o eliminan las reglas de la base de conocimiento. Para un proyecto estudiantil, esto podría ser simplemente editar un archivo de texto o código.
* **Módulo de Explicación (Explanation Module):** (Deseable) Permite al sistema explicar por qué llegó a cierto diagnóstico. "Llegué a esta conclusión porque usted indicó X, Y y Z, lo que activa la regla W."

**3. Tecnología/Herramientas Sugeridas:**

* **Lenguajes de Programación:**
  + **Python:** Excelente opción por su simplicidad, gran cantidad de librerías y frameworks. Hay librerías para lógica de reglas (ej. clipsy, pyknow, o puedes implementarlo tú mismo).
  + **Prolog:** Diseñado específicamente para lógica y programación declarativa, lo que lo hace ideal para sistemas basados en reglas. Podría ser un buen desafío de aprendizaje.
  + **Java/C#:** También viables, pero pueden requerir más código para la implementación del motor de inferencia si no usas un framework.
* **Frameworks/Librerías (Python):**
  + Puedes implementar un motor de inferencia básico tú mismo con diccionarios y listas para las reglas.
  + Explora librerías como pyknow o incluso Clipsy (que es una interfaz para CLIPS, un sistema experto robusto).
* **Interfaz de Usuario:**
  + **Consola:** La forma más sencilla para un prototipo inicial.
  + **GUI (Graphical User Interface):** Tkinter, PyQt, Kivy (Python) para una interfaz más amigable.
  + **Web:** Flask o Django (Python) para una interfaz web, lo que permitiría el acceso desde cualquier navegador.

**4. Pasos para el Desarrollo del Proyecto:**

1. **Definir el Alcance:** Elige 5-10 problemas específicos de hardware/software que tu sistema va a diagnosticar.
2. **Recopilar Conocimiento:** Investiga los síntomas, causas y soluciones de esos problemas. Puedes consultar foros técnicos, manuales, etc.
3. **Diseñar la Base de Conocimiento:** Traduce el conocimiento recopilado en reglas IF-THEN. Intenta ser lo más exhaustivo posible para los problemas seleccionados.
4. **Implementar el Motor de Inferencia:** Decide si será encadenamiento hacia adelante o hacia atrás (o una combinación). Codifica la lógica que procesará las reglas y los hechos.
5. **Desarrollar la Interfaz de Usuario:** Cómo el sistema hará preguntas al usuario y mostrará los resultados.
6. **Pruebas:** ¡Crucial! Prueba el sistema con escenarios reales y ficticios para asegurarte de que diagnostica correctamente y maneja casos donde no tiene suficiente información.
7. **Refinamiento y Expansión:** Una vez que funcione lo básico, puedes añadir más problemas, mejorar las explicaciones o hacer la interfaz más robusta.

**5. Consideraciones Adicionales:**

* **Incertidumbre:** ¿Cómo manejarás situaciones donde la información es ambigua o el usuario no está seguro? Para un proyecto inicial, puedes omitir esto, pero los sistemas expertos reales a menudo usan factores de certeza.
* **Manejo de "No sé":** Si el usuario no sabe la respuesta a una pregunta, ¿qué hace el sistema?
* **Bucles Infinitos:** Asegúrate de que tu motor de inferencia no caiga en bucles al procesar las reglas.
* **Actualización de la Base de Conocimiento:** Cómo se mantendría el sistema actualizado con nuevos problemas o soluciones.

¡Este es un proyecto muy gratificante porque verás cómo los principios de la inteligencia artificial se aplican a un problema real y tangible! ¡Mucha suerte!

**🔧 Componentes del Sistema Experto**

**Base de Conocimiento**

* **Síntomas categorizados** : alimentación, arranque, pantalla, rendimiento, sonidos, temperatura, cambios recientes
* **Reglas de diagnóstico** : 8 reglas principales que cubren problemas comunes de hardware y software
* **Problemas diagnosticables** :
  + Falla en fuente de poder
  + Problemas de placa madre
  + Fallas de memoria RAM
  + Problemas de disco duro
  + Sobrecalentamiento de la CPU
  + Problemas de tarjeta gráfica
  + Corrupción de software
  + Conflictos de conductores

**Motor de Inferencia**

* **Encadenamiento hacia adelante** : evalúa síntomas para llegar a diagnósticos
* **Sistema de coincidencias** : calcular porcentajes de coincidencia entre síntomas y reglas
* **Ranking por confianza** : ordena diagnósticos por probabilidad
* **Umbral de decisión** : 60% mínimo de coincidencia para considerar un diagnóstico válido

**Interfaz de usuario**

* **Cuestionario interactivo** : guía paso a paso con preguntas específicas
* **Visualización de progreso** : barra que muestra el avance del diagnóstico
* **Resultados detallados** : problema identificado, descripción y soluciones
* **Motor de reglas visible** : muestra qué reglas están siendo evaluadas en tiempo real

**Características Técnicas**

* **Múltiples diagnósticos** : puede sugerir hasta 3 problemas posibles
* **Soluciones prácticas** : pasos específicos para resolver cada problema
* **Sistema de confianza** : porcentaje de certeza para cada diagnóstico
* **Reinicio fácil** : permite hacer nuevos diagnósticos

**🎯 Aplicación Práctica**

Este sistema demuestra conceptos clave de sistemas expertos:

* **Representación del conocimiento** mediante reglas SI-ENTONCES
* **Inferencia automatizada** basada en síntomas
* **Explicación de razonamiento** mostrando reglas activas
* **Gestión de incertidumbre** con porcentajes de confianza

El sistema es especialmente útil para técnicos, usuarios domésticos y estudiantes que quieran entender cómo funciona el diagnóstico automatizado de problemas informáticos.