1. **¿Qué es la herencia y cuál es su principal objetivo en la programación orientada a objetos?**
   La herencia es un mecanismo en programación orientada a objetos que permite a una clase (subclase o clase derivada) heredar atributos y métodos de otra clase (superclase o clase base). El principal objetivo de la herencia es promover la reutilización de código y permitir la creación de jerarquías de clases, donde las clases derivadas pueden extender o especializar el comportamiento de las clases base.

2. **¿Cuál es la importancia de la encapsulación en el contexto de la herencia?**
   La encapsulación permite ocultar los detalles internos de una clase y proteger sus datos, de manera que solo métodos específicos puedan acceder o modificar sus atributos. En el contexto de la herencia, la encapsulación ayuda a controlar qué información de la clase base se hereda y cómo se utiliza en las subclases, asegurando que solo se exponga lo necesario y manteniendo la integridad de los datos.

3. **Describe los tipos de visibilidad (public, private, protected) en el contexto de herencia.**

   - **public**: Los miembros públicos de una clase son accesibles desde cualquier otra clase, incluidas las clases derivadas.
   - **protected**: Los miembros protegidos son accesibles dentro de la clase y en sus subclases, pero no desde fuera de la jerarquía de herencia.
   - **private**: Los miembros privados solo son accesibles dentro de la propia clase. No son accesibles desde subclases ni otras clases, aunque pueden heredarse, pero sin acceso directo.

4. **¿Qué es una especialización?**
   La especialización es el proceso mediante el cual una subclase se deriva de una clase base para añadir funcionalidades o comportamientos específicos, convirtiéndose en una versión especializada de la clase base. Esto permite a la subclase diferenciarse con características adicionales o modificadas.

5. **¿Cuáles son las ventajas y los posibles problemas que pueden surgir al utilizar la herencia múltiple?**

   - **Ventajas**: Permite que una clase herede características de múltiples clases, proporcionando una mayor flexibilidad y reusabilidad del código.
   - **Problemas**: La herencia múltiple puede llevar a conflictos, como la ambigüedad cuando una subclase hereda métodos o atributos con el mismo nombre de diferentes clases base. Esto puede complicar el diseño y el mantenimiento del código.

B. Análisis y Ejecución del Código en C++

1. **Explicación del funcionamiento del programa:**
   El programa define una jerarquía de clases que representan diferentes tipos de personas. La clase base Persona almacena el nombre y la edad de una persona y tiene un método mostrarPersona para mostrar esta información. La clase Empleado hereda

de Persona y añade un atributo sueldo, junto con el método mostrarEmpleado que muestra el nombre, edad, y sueldo. La clase Estudiante también hereda de Persona y añade un atributo notaFinal con su respectivo método mostrarEstudiante. Finalmente, la clase Tecnico hereda de Estudiante, añade el atributo esp ecialidad, y tiene el método mostrarTecnico para mostrar toda la información de un técnico.

2. **Completar los comentarios indicados con líneas punteadas:**

```
class Persona {
   private: // Atributos de Persona
      string nombre;
      int edad;
   public: // Constructor de Persona
      Persona(string, int);
      void mostrarPersona(); // Método para mostrar datos de Persona
};
class Empleado : public Persona {
   private: // Atributo de Empleado
      float sueldo;
   public: // Constructor de Empleado
      Empleado(string, int, float);
      void mostrarEmpleado(); // Método para mostrar datos de Empleado
};
class Estudiante : public Persona {
   private: // Atributo de Estudiante
      float notaFinal;
   public: // Constructor de Estudiante
      Estudiante(string, int, float);
      void mostrarEstudiante(); // Método para mostrar datos de Estudiante
};
class Tecnico : public Estudiante {
   private: // Atributo de Tecnico
      string especialidad;
   public: // Constructor de Tecnico
```

```
    Tecnico(string, int, float, string);

    void mostrarTecnico(); // Método para mostrar datos de Tecnico

};
```

¿Cuál es el nombre de la clase padre en la jerarquía?
La clase padre en la jerarquía es Persona.

¿Qué clase hereda de la clase Persona?
Las clases Empleado y Estudiante heredan de Persona.

¿Cuál es la relación entre las clases Estudiante y Tecnico?
Tecnico es una subclase de Estudiante, lo que significa que Tecnico hereda de Estudiante.

¿Cuál es el propósito del constructor de Empleado?
El propósito del constructor de Empleado es inicializar el nombre y la edad del empleado a través del constructor de Persona, y despues inicializar el atributo específico sueldo de la clase Empleado.

¿Cuál es el propósito del método mostrarPersona()?
El propósito de mostrarPersona() es mostrar los atributos nombre y edad de la clase Persona. Este método es reutilizado por las clases derivadas para mostrar información común.

¿Qué método se utiliza para mostrar el salario del empleado?
El método mostrarEmpleado() se us apara mostrar el salario del empleado, ya que este método llama a mostrarPersona() y luego muestra el atributo sueldo.