	Excelente	Bien	Suficiente
Elaboración del AOO	Se aporta un estudio original y completo de detección de posibles clases a partir de los enunciados mediante el método de análisis sintáctico de Coad-Yourdon. Se aporta un diagrama conceptual del sistema a construir correcto, indicando las entidades principales detectadas, su relación y localización (fuera o dentro del sistema). Las clases/entidades detectadas en el AOO son correctas y suficientes para el problema planteado.	Se aporta un diagrama conceptual del sistema a construir correcto, indicando las entidades principales detectadas, su relación y localización (fuera o dentro del sistema). Las clases/entidades detectadas en el AOO son correctas ysuficientes para el problema planteado.	Se aporta un diagrama conceptual del sistema a construir correcto, indicando las entidades principales detectadas. Las clases/entidades detectadas en el AOO son correctas en su mayoría, pero no suficientes, para el problema planteado.
Elaboración del DOO	Se aporta un diagrama de clases UML que representa la estructura base del sistema. Se aportan diagramas (secuencia, actividades o estado) que modelan el comportamiento y la interacción de las principales clases y objetos del sistema. Se usa la herramienta de modelado UML recomendada para la generación de los diagramas UML. Se aporta la descripción de los contratos de operaciones. Las clases elaboradas en el DOO son correctas para el problema planteado. Se indican correctamente sus responsabilidades y colaboraciones. En general, las clases y métodos presentan un alto grado de cohesión.	Se aporta un diagrama de clases UML que representa la estructura base del sistema. Se aportan diagramas (secuencia, actividades o estado) que modelan el comportamiento y la interacción de algunas clases y objetos del sistema. Se usa la herramienta de modelado UML recomendada para la generación de los diagramas UML. Se aporta la descripción de la mayoría de los contratos de operaciones, al menos, de las operaciones que conforman el interfaz público de las clases. Las clases elaboradas en el DOO son correctas para el problema planteado. Se indican correctamente sus responsabilidades y colaboraciones.	Se aporta un diagrama de clases UML que representa la estructura base del sistema. Se usa la herramienta de modelado UML recomendada para la generación de los diagramas UML . Se aporta la descripción de algunos contratos de operaciones. La mayoría de clases elaboradas en el DOO son correctas para el problema planteado. Se indican sus responsabilidades y colaboraciones.
Aplicación de los conceptos: clases, encapsulación y composición	Se definen las clases necesarias para el sistema, especificando correctamente su interfaz funcional público. Todas las clases aplican correctamente el principio de encapsulación sin exponer en su interfaz funcional público detalles de implementación de sus datos. Se utiliza adecuadamente la composición en la definición de las clases del sistema.	Se definen las clases necesarias para el sistema, especificando correctamente su interfaz funcional público. La mayoría de las clases aplican correctamente el principio de encapsulación sin exponer en su interfaz funcional público detalles de implementación de sus datos. Se utiliza la composición en la definición de las clases del sistema.	Se definen las clases necesarias para el sistema, especificando su interfaz funcional público. Las clases más importantes en el proyecto aplican el principio de encapsulación sin exponer en su interfaz funcional público detalles de implementación de sus datos o exponiendo mínimos detalles de implementación de sus datos. Se utiliza la composición en la definición de las clases del sistema.
Aplicación de los conceptos: herencia, polimorfismo	Se definen todas las jerarquías de clases necesarias para el sistema de manera correcta. Se define correctamente el interfaz funcional público de la clase base de la jerarquía para poder	Se definen todas las jerarquías de clases necesarias para el sistema de manera correcta. Se define correctamente el interfaz funcional público de la clase base de la jerarquía para poder	Se definen todas las jerarquías de clases necesarias para el sistema de manera correcta. Se define correctamente el interfaz funcional público de la clase base de la jerarquía para

	sacar provecho del polimorfismo. Se definen todos los algoritmos polimórficos y todas las estructuras de datos polimórficas necesarias para el sistema, sacando el máximo provecho del polimorfismo y evitando en todos los casos la repetición de código.	sacar provecho del polimorfismo. Se definen todos los algoritmos polimórficos y todas las estructuras de datos polimórficas necesarias para el sistema.	poder sacar provecho del polimorfismo. Se definen la mayoría de algoritmos polimórficos y estructuras de datos polimórficas necesarias para el sistema.
Aplicación de los patrones de diseño	Se aplican correctamente todos los patrones de diseño estudiados en todos los casos posibles. No se detecta ningún antipatrón en el código entregado.	Se aplican correctamente algunos de los patrones de diseño estudiados en todos los casos posibles. Se detecta un antipatrón en el código entregado.	Se aplican correctamente algunos de los patrones de diseño estudiados en los casos básicos. Se detectan varios antipatrones en el código entregado.
Aplicación de los conceptos de programación genérica	Se definen correctamente las plantillas de clase y función especificadas en los enunciados, junto a su especialización para punteros. Se definen plantillas de clases no especificadas en el enunciado, que mejoran ostensiblemente el código del sistema y cuyo uso está perfectamente razonado.	Se definen correctamente las plantillas de clase y función especificadas en los enunciados, junto a su especialización para punteros. Se definen plantillas de clases no especificadas en el enunciado, que mejoran el código del sistema.	Se definen correctamente las plantillas de clase y función especificadas en los enunciados, junto a su especialización para punteros.
Uso del mecanismo de excepciones	Se utilizan correctamente excepciones, de forma que se asegura el contrato de todas las rutinas del proyecto.	Se utilizan correctamente excepciones, de forma que se asegura el contrato de las rutinas más importantes del proyecto y de métodos constructores y destructores de clases.	Se utilizan correctamente excepciones, de forma que se asegura el contrato de algunas de las rutinas más importantes del proyecto.
Uso de la STL	Se utilizan correctamente distintos tipos de contenedores y algoritmos de la STL, utilizando los más apropiados en cada caso, dependiendo de sus características. Se hace un uso avanzado de la definición de objetos función para la personalización del comportamiento de los contenedores y algoritmos de la STL.	Se utilizan correctamente algunos tipos de contenedores y algoritmos de la STL, utilizando los más apropiados en cada caso, dependiendo de sus características. Se hace un buen uso de la definición de objetos función para la personalización del comportamiento de los contenedores y algoritmos de la STL.	Se utiliza correctamente algunos tipos de contenedores y algoritmos de la STL. Se hace un uso básico de la definición de objetos función para la personalización del comportamiento de los contenedores y algoritmos de la STL.
Pruebas unitarias del sistema	Se ha elaborado un completo caso de pruebas (Test Case) por cada una de las clases desarrolladas, aportando una prueba de funcionamiento por cada método miembro público de la clase y por cada proceso principal de la misma (secuencias de uso de métodos).	Se ha elaborado un caso de pruebas (Test Case) por cada una de las clases desarrolladas, aportando al menos una prueba de funcionamiento del proceso principal de la misma (secuencias de uso de métodos).	Se han elaborado casos de pruebas (Test Case) para las clases principales (en cuanto a importancia del comportamiento implementado) desarrolladas, aportando al menos una prueba de funcionamiento del proceso principal de la misma (secuencias de uso de métodos).
Gestión de la memoria dinámica	El sistema gestiona correctamente la memoria dinámica utilizada de manera que no se producen pérdidas de memoria dinámica ni se detecta ningún tipo de error de uso de la memoria mediante la herramienta Valgrind.	El sistema gestiona correctamente la memoria dinámica utilizada de manera que no se producen pérdidas de memoria dinámica, aunque se detecta algún error de uso de la memoria mediante la herramienta Valgrind.	El sistema puede presentar pequeñas pérdidas de memoria dinámica y se detecta algún error de uso de la memoria mediante la herramienta Valgrind.
Funcionamiento del sistema	El sistema funciona siempre correctamente según las especificaciones dadas en el enunciado, dentro	El sistema funciona siempre correctamente según las especificaciones dadas en el enunciado, aunque	El sistema funciona al menos en las ocasiones propuestas como ejemplos en el enunciado o

	de unos parámetros de tiempo y uso de recursos adecuados. El sistema controla todos los posibles errores de funcionamiento, avisando al usuario correctamente de su causa.	puede consumir demasiado tiempo y recursos. El sistema controla la mayoría de errores posibles de funcionamiento, y avisa en la mayoría de ocasiones al usuario de su causa.	demos proporcionadas, atendiendo a las especificaciones dadas en el enunciado; aunque puede consumir demasiado tiempo y recursos. El sistema controla los errores más importantes de funcionamiento, y avisa en algunas ocasiones al usuario de su causa.
Documentación interna del código	Los atributos de todas las clases poseen un comentario en formato doxygen indicando su misión. Los métodos de todas las clases poseen un comentario en formato Doxygen indicando su misión, la misión de sus parámetros, los valores de retorno y las excepciones que lanza. Se indica, además, el contrato de operación mediante la especificación de precondiciones y poscondiciones. En los algoritmos más complejos se indica también su complejidad. Permite su extracción en formato HTML mediante la herramienta doxygen sin producir ningún error.	Los atributos de todas las clases poseen un comentario en formato doxygen indicando su misión. Los métodos de todas las clases poseen un comentario en formato Doxygen indicando su misión, la misión de sus parámetros, los valores de retorno y las excepciones que lanza. Se indica, en la mayoría de algoritmos, el contrato de operación mediante la especificación de precondiciones y poscondiciones y su complejidad. Permite su extracción en formato HTML mediante la herramienta doxygen sin producir ningún error.	La mayoría de atributos y métodos poseen un comentario en formato doxygen indicando su misión. Se indica, en los algoritmos más importantes, el contrato de operación mediante la especificación de precondiciones y poscondiciones y su complejidad. Permite su extracción en formato HTML mediante la herramienta doxygen sin producir ningún error o produciendo algún pequeño error.
Documentación externa	Se entrega una documentación externa del proyecto de programación realizado que cumple estrictamente con las especificaciones establecidas de formato, organización y contenidos. No contiene faltas de ortografía. Realiza una exposición completa, clara y concisa de los contenidos.	Se entrega una documentación externa del proyecto de programación realizado que cumple con las especificaciones establecidas de formato, organización y contenidos. No contiene faltas de ortografía. Realiza una exposición completa de los contenidos.	Se entrega una documentación externa del proyecto de programación realizado que cumple generalmente con las especificaciones establecidas de formato, organización y contenidos. No contiene faltas de ortografía evidentes. Realiza una exposición suficiente de los contenidos.
Estilo de código	El código entregado cumple todas las especificaciones del estilo de código utilizado en la asignatura, como son: correcta identación de todo el código, uso de un único estilo en los identificadores de tipo, método y variable, comentarios, nombres de ficheros, declaraciones de funciones, bucles, especificación de condiciones, inicialización de variables, directivas del preprocesador y macros.	El código entregado cumple la mayoría de las especificaciones del estilo de código utilizado en la asignatura, como son: correcta identación de todo el código, uso de un único estilo en los identificadores de tipo, método y variable, comentarios, nombres de ficheros, declaraciones de funciones, bucles, especificación de condiciones, inicialización de variables, directivas del preprocesador y macros.	El código entregado cumple las especificaciones más importantes del estilo de código utilizado en la asignatura, como son: correcta identación de todo el código, uso de un único estilo en los identificadores de tipo, método y variable.