Name & Title: - Archival Material Management System

Introduction :- An Archival Material Management System is a software application designed to help organisations manage their archival materials, such as documents, photographs, audio recordings, and video recordings. The system provides tools for organising and categorising materials, searching for specific items, and tracking the status of materials throughout the archival process. It may also include features for digitising and preserving materials for long-term storage and accessibility. An effective Archival Material Management System can help organisations maintain accurate records, streamline workflows, and ensure the preservation of valuable historical materials for future generations.

Enums Used:

Material Type:

Enum Name	Description
MaterialType	An enumeration of different types of materials.
BOOK	Refers to a book.
JOURNAL	Refers to a journal.
NEWSPAPER	Refers to a newspaper.

Book Type:

Enum Name	Description
BookType	An enumeration of different types of books.
NOVEL	A type of book that is fictional and typically longer in length.
BIOGRAPHY	A type of book that is non-fictional and focuses on the life of a specific person.
HISTORY	A type of book that is non-fictional and focuses on past events, people, and societies.

Journal Type:

Enum Name	Description
JournalType	An enumeration of different types of journals.
SCIENCE	A journal type for scientific articles.
LITERATURE	A journal type for literary articles.
ART	A journal type for articles related to the arts.

Newspaper Type:

Enum Name	Description
NewspaperType	An enumeration of different types of newspapers.
DAILY	Represents a daily newspaper.
WEEKLY	Represents a weekly newspaper.
MONTHLY	Represents a monthly newspaper.

Union Used:

Material Details:

Name	Data Type	Description
MaterialDetails	Union	A union data type that contains three different struct types: book, journal, and newspaper.
book	Struct	A struct data type that contains three members: pages (an integer representing the number of pages in the book), author (a character array representing the name of the author of the book), and type (an enum representing the type of book).
pages	Integer	An integer representing the number of pages in the book.
author	Character array	A character array representing the name of the author of the book.
type	Enum	An enum representing the type of book (e.g. fiction, non-fiction, biography, etc.).
journal	Struct	A struct data type that contains three members: issue (an integer representing the issue number of the journal), publisher (a character array representing the name of the publisher of the journal), and type (an enum representing the type of journal).
issue	Integer	An integer representing the issue number of the journal.
publisher	Character array	A character array representing the name of the publisher of the journal.
type	Enum	An enum representing the type of journal (e.g. scientific, medical, academic, etc.).
newspaper	Struct	A struct data type that contains two members: editor (a character array representing the name of the editor of the newspaper) and type (an enum representing the type of newspaper).
editor	Character array	A character array representing the name of the editor of the newspaper.

type	Enum	An enum representing the type of newspaper (e.g. daily, weekly, etc.).

Structures Used:

Material:

Property	Data Type	Description
title	char[50]	The title of the material.
type	enum MaterialType	The type of the material, defined by the MaterialType enum.
details	union MaterialDetails	A union containing details specific to the material type.

Archive:

Property	Data Type	Description
materials	struct Material[100]	An array of Material structs with a maximum size of 100.
count	int	An integer representing the number of Material structs currently stored in the materials array.

Problem Description:

The following functions will be used for the assignment:

Function Name	Parameter(s)	Return Type	Description
addMaterial	struct Archive *archive, struct Material material	int	Adds a new material to the archive. Returns 0 if successful, -1 if archive is full or material title already exists.
findMaterial	struct Archive *archive, char *title	struct Material *	Finds a material in the archive with the given title. Returns a pointer to the material if found, NULL otherwise.
filterMaterials	struct Archive *archive, enum MaterialType type	void	Filters materials in the archive by type and creates a new archive containing only materials of the specified type.
updateMaterial	struct Archive *archive, char *title, union MaterialDetails details	int	Updates the details of a material in the archive with the given title. Returns 0 if successful, -1 if archive or title is null, -2 if material is not found, -3 if book type is invalid, -4 if journal type is invalid, -5 if newspaper type is invalid, -6 if material type is invalid.

removeMaterial	struct Archive *archive, char *title	void	Removes a material from the archive with the given title. Does nothing if the material is not found.
filterMaterialsByAuthor	struct Archive *archive, char *author	struct Material*	This function takes a pointer to an Archive struct and a string of the author's name. It searches the archive for any material whose author matches the given name and returns an array of matching materials.

Test Cases :

Test Case Name	Input	Output	Explanation
testAddMaterialSuccess	A struct Archive archive is initialised to contain 0 materials. A struct Material material is initialised with the title "Book Title", type BOOK, and a book struct. addMaterial function is called with archive and material as parameters.	The addMaterial function returns 0 indicating that the material was successfully added to the archive. The archive count is incremented by 1.	This test case checks the addMaterial function's ability to add a material to an empty archive. The test case verifies that addMaterial function returns 0, which indicates that the material was successfully added to the archive, and that the archive count is incremented by 1.
testAddMaterialDuplicateTitle	A struct Archive archive is initialised to contain 0 materials. Two struct Materials, material1 and material2, are initialised with the same title "Book Title", but different types and details. addMaterial function is called twice, once with archive and material1 as parameters and once with archive and material2 as parameters.	The first call to addMaterial function returns 0 indicating that the material1 was successfully added to the archive. The second call to addMaterial function returns -1 indicating that the material2 was not added to the archive due to a duplicate title. The archive count is incremented by 1, but only one material is added to the archive.	This test case checks the addMaterial function's ability to handle adding a material with a duplicate title to an archive. The test case verifies that the first material is added to the archive successfully, and the second material is not added due to the duplicate title. The archive count should be incremented by 1, but only one material is added to the archive.
testAddMaterialDifferentTypes	A struct Archive archive is initialised to contain 0 materials. Three struct Materials, book, journal, and	All three calls to addMaterial function return 0 indicating that all three materials were successfully added to	This test case checks the addMaterial function's ability to add multiple materials with different types and details to an

	newspaper, are initialised with different types and details. addMaterial function is called three times, once with archive and book as parameters, once with archive and journal as parameters, and once with archive and newspaper as parameters.	the archive. The archive count is incremented by 3.	archive. The test case verifies that all three materials are added to the archive successfully, and the archive count is incremented by 3
testFindMaterialEmptyArchive	Empty Archive	NULL	This test checks if findMaterial() returns NULL when searching for a material that is not in the archive. It initialises an empty archive and searches for the book "The Catcher in the Rye". The function should return NULL since the archive is empty.
testFindMaterialNotFound	Archive with some materials	NULL	This test checks if findMaterial() returns NULL when searching for a material that is not in the archive. It initialises an archive with three books and searches for the book "The Catcher in the Rye". The function should return NULL since this book is not in the archive.
testFindMaterialNotFound2	Archive with one material	NULL	This test checks if findMaterial() returns NULL when searching for a material that is not in the archive. It initialises an archive with one book and searches for the book "The Catcher in the Rye". The function should return NULL since this book is not in the archive.
testFilterMaterials_NoMatch	Archive with three materials	Archive with three materials	This test checks if filterMaterials() returns an archive with the same number of

			materials when there is no match. It initialises an archive with one book, one journal, and one newspaper. It filters the archive for materials of type NEWSPAPER. Since there are no newspapers in the archive, the function should return an archive with the same three materials.
testFilterMaterials_Match	Archive with three materials	Archive with one material	This test checks if filterMaterials() returns an archive with only the materials that match the filter. It initialises an archive with one book, one journal, and one newspaper. It filters the archive for materials of type JOURNAL. Since there is only one journal in the archive, the function should return an archive with only one material.
testFilterMaterials2	Archive with three materials	Archive with one material	This test checks if filterMaterials() returns an archive with only the materials that match the filter. It initialises an archive with one book, one journal, and one newspaper. It filters the archive for materials of type JOURNAL. Since there is only one journal in the archive, the function should return an archive with only one material, the journal called "Nature".
testUpdateMaterialNullTitle	Archive with one material	NULL	This test checks if updateMaterial() returns NULL when the title is NULL. It initialises an archive with one book and

			attempts to update the book's title to NULL. Since the title cannot be NULL, the function should return NULL
testUpdateMaterialSuccess	The archive contains one book titled "The Great Gatsby". The new author's name is "Francis Scott Fitzgerald". The new number of pages is 218. The new book type is "HISTORY". The material title is "The Great Gatsby".	0	This test case is checking whether the updateMaterial() function updates the details of a material successfully. First, a book with the title "The Great Gatsby" is added to the archive. Then, the book is updated with new details, such as the author name, number of pages, and book type. The function is expected to return 0 to indicate that the update was successful.
testUpdateMaterialNullArchive	The archive is null. The material title is "The Great Gatsby".	-1	This test case is checking whether the updateMaterial() function handles null archives correctly. Since the archive is null, the function should return -1 to indicate an error.
testRemoveMaterialSuccess1	The archive contains one book titled "The Great Gatsby". The material title is "The Great Gatsby".	None	This test case is checking whether the removeMaterial() function removes a material successfully when the archive contains only one material. First, an archive with one material is initialised. Then, the material is removed by passing its title to the removeMaterial() function. After the removal, the archive count should be 0, indicating that the material was removed successfully.
testRemoveMaterialSuccess2	The archive contains two books: "The Great Gatsby" and	None	This test case is checking whether the removeMaterial()

	"To Kill a Mockingbird". The material title is "To Kill a Mockingbird".		function removes a material successfully when the archive contains two materials. First, an archive with two materials is initialised. Then, one of the materials is removed by passing its title to the removeMaterial() function. After the removal, the archive count should be 1, indicating that the material was removed successfully.
testRemoveMaterialSuccess3	The archive contains three books: "The Great Gatsby", "To Kill a Mockingbird", and "The Sun Also Rises". The material title is "The Great Gatsby".	None	This test case is checking whether the removeMaterial() function removes a material successfully when the archive contains three materials. First, an archive with three materials is initialised. Then, the first material is removed by passing its title to the removeMaterial() function. After the removal, the archive count should be 2, indicating that the material was removed successfully.
testAuthorNotFound	The archive contains two materials: a book with author "Stephen Hawking" and a journal with publisher "John Wiley & Sons". The author is "John Doe".	NULL	This test case is checking whether the filterMaterialsByAuthor() function correctly handles cases where the author is not found in the archive. Since "John Doe" is not an author of any material in the archive, the function should return NULL.
testOneMatchingBook	The archive contains two books: "A Brief History of Time" by "Stephen Hawking" and "A Tale of Two Cities" by "Charles Dickens". The author	A pointer to a material with title "A Brief History of Time", author "Stephen Hawking", and type "NOVEL".	This test case is checking whether the filterMaterialsByAuthor() function correctly filters materials by author. Since "Stephen Hawking" is the author of one of

	is "Stephen Hawking".		the books in the archive, the function should return a pointer to that book
testFilterMaterialsByAuthor_MatchingBooks	archive: a struct containing an array of 3 materials with their details and count, author: a string variable representing the name of an author	result: a pointer to an array of materials filtered by the given author	This test case verifies the correctness of the filterMaterialsByAuthor function by checking if it returns an array of materials that match the given author's name. The test case passes the archive variable and author variable to the function and checks if the returned array of materials matches the expected result. It also verifies the returned materials' properties, such as their type, pages, and issue number.