Name & Title: - Animal Adoption Management System

Introduction :- The Animal Adoption Management System is a software application designed to streamline the process of animal adoption in shelters and rescue organisations. The system helps to manage the adoption process from start to finish, including tracking animal intake, medical history, vaccinations, and adoption applications. It also helps to manage the communication and coordination between staff, volunteers, and potential adopters. The system can improve the overall efficiency of animal adoption programs, increase the chances of successful adoptions, and ultimately help more animals find their forever homes.

Macro’s Used:

Macro	Value	Nuances
MAX_AGE	20	Defines the maximum age of an animal in the system.
DEFAULT_AGE	1	Defines the default age of an animal if not specified.
MAX_ANIMALS	100	Defines the maximum number of animals in the system.
MAX_NAME_LENGTH	50	Defines the maximum length of an animal's name.
MAX_TYPE_LENGTH	10	Defines the maximum length of an animal's type.

Enums Used:

Animal Type:

Enum Name	Description
CAT	A constant representing a type of animal (cat)
DOG	A constant representing a type of animal (dog)
BIRD	A constant representing a type of animal (bird)
MOUSE	A constant representing a type of animal (mouse)

Function Status:

Enum Name	Description
SUCCESS	A constant value that represents a successful function execution.

FAILURE	A constant value that represents a failed function execution.
INVALID_INPUT	A constant value that represents an invalid input parameter to a function.

Structures Used:

Animal Count:

Property	Data Type	Description
count	int	An integer representing the number of animals of a specific type.
type	AnimalType	An enumerated data type representing the type of animal (e.g., dog, cat, bird, etc.).

Animal:

Property	Data Type	Description
name	char[50]	The name of the animal, represented as a character array with a maximum length of 50 characters.
age	int	The age of the animal, represented as an integer value.
type	AnimalType	The type of animal, represented by an enum type called "AnimalType". The possible values of this type will depend on how the enum is defined elsewhere in the code.

Problem Description:

The following functions will be used for the assignment:

Function Name	Parameters	Return Type	Explanation
addAnimal	Animal *animals (pointer to an array of Animal structures), int *numAnimals (pointer to an integer)	void	This function adds a new animal to the array of animals. It checks if there is enough space to add a new animal, initializes the new animal, gets the animal's name, age, and type, and adds the new animal to the array.
searchAnimal	Animal *animals (pointer to an array of Animal structures), int numAnimals (an integer), const char *name (a pointer to a string)	Animal * (pointer to an Animal structure)	This function searches for an animal in the array by name and returns a pointer to its details. If the animal is not found, it returns NULL.
deleteAnimal	Animal *animals (pointer to an array of Animal	bool	This function deletes an animal from the array by name. It checks if

	structures), int *numAnimals (pointer to an integer), const char *nameToDelete (a pointer to a string)		there are any animals in the array, finds the index of the animal with the given name, shifts all animals after the deleted animal back by one index, decrements the number of animals in the array, and returns true if the animal was successfully deleted, or false otherwise.
updateAnimal	Animal *animals (pointer to an array of Animal structures), int numAnimals (an integer), char *name (a pointer to a string), int age (an integer), AnimalType type (an enumeration)	FunctionStatus (an enumeration)	This function updates the details of an animal in the array by name, age, and type. It checks for valid input parameters, searches for the animal to update, and updates the animal's details.
sortAnimals	Animal *animals (pointer to an array of Animal structures), int numAnimals (an integer), char sortBy (a character)	Animal * (pointer to an Animal structure)	This function sorts the array of animals by the given attribute (name, age, or type) and returns a pointer to the sorted array. It checks for invalid sortBy parameter, allocates memory for a copy of the array, copies the contents of the original array into the copy, and sorts the copy by the given attribute.
countAnimalsByType	Animal *animals, int numAnimals	AnimalCount*	This function takes an array of Animal structures and the number of animals in the array as input parameters. It then counts the number of animals of each type (CAT, DOG, and BIRD) and returns a dynamically allocated array of AnimalCount structures. Each AnimalCount structure contains the count of animals for a specific type.

Test Cases :

Test Case Name	**Input**	**Output**	**Explanation**
testAddAnimalWithDefaultValues	none	numAnimals = 1, name = "Unknown", age = 1, type = CAT	The function addAnimal is being tested to check if it can add an animal with default values in the Animal array.
testAddAnimalWithName	name = "Fluffy", age = 1, type = CAT	numAnimals = 1, name = "Fluffy", age = 1, type = CAT	The function addAnimal is being tested to check if it can add an animal with a specific name in the Animal array.

testAddInvalidAge	none	numAnimals = 2, age = MAX_AGE+1 for the first animal, age = -1 for the second animal	The function addAnimal is being tested to check if it can handle the case where the age of an animal is greater than MAX_AGE or less than 0.
testSearchAnimalValid	animal array with an animal named "Tom"	a pointer to the animal named "Tom", age = 2, type = CAT	The function searchAnimal is being tested to check if it can find an animal in the Animal array by its name.
testSearchAnimalInvalid	animal array without an animal named "Snoopy"	nullptr	The function searchAnimal is being tested to check if it can handle the case where the animal being searched for is not in the Animal array.
testSearchAnimalMultipleMatches1	animal array with two animals named "Fluffy"	a pointer to the first animal named "Fluffy", age = 2, type = CAT	The function searchAnimal is being tested to check if it can handle the case where there are multiple animals with the same name in the Animal array.
testDeleteAnimalExisting	animal array with an animal named "Rufus"	numAnimals = 2, animals[0] = {"Fluffy", 2, CAT}, animals[1] = {"Tweety", 3, BIRD}, return value = true	The function deleteAnimal is being tested to check if it can delete an animal that exists in the Animal array.
testDeleteAnimalNonExisting	animal array without an animal named "Misty"	numAnimals = 3, animals[0] = {"Fluffy", 2, CAT}, animals[1] = {"Rufus", 1, DOG}, animals[2] = {"Tweety", 3, BIRD}, return value = false	The function deleteAnimal is being tested to check if it can handle the case where the animal being deleted is not in the Animal array.
testDeleteAnimalEmptyArray	Animal Array = {}, numAnimals = 0, name = "Fluffy"	result = false, numAnimals = 0	Tests whether the function can correctly return false and not modify the array if an animal is deleted from an empty array.
testUpdateAnimalValid	Animal Array = {{"Fluffy", 2, CAT}, {"Rex", 3,	result = SUCCESS, animals[0].age = 4, animals[0].type = BIRD	Tests whether the function can update an existing animal in the

	DOG}}; numAnimals = 2, name = "Fluffy", age = 4, type = BIRD		array with valid input parameters. It checks that the returned value is SUCCESS and that the animal's age and type have been correctly updated.
testUpdateAnimalInvalidAge	Animal Array = {{"Fluffy", 2, CAT}, {"Rex", 3, DOG}}; numAnimals = 2, name = "Fluffy", age = -1, type = BIRD	result = INVALID_INPUT, animals[0].age = 2	Tests whether the function can return INVALID_INPUT and not update the animal's age if an invalid age is provided as input.
testUpdateAnimalInvalidType	Animal Array = {{"Fluffy", 2, CAT}, {"Rex", 3, DOG}}; numAnimals = 2, name = "Fluffy", age = 4, type = MOUSE	result = INVALID_INPUT, animals[0].type = CAT	Tests whether the function can return INVALID_INPUT and not update the animal's type if an invalid type is provided as input.
testSortAnimalsByName	Animal Array = {{"Charlie", 3, CAT}, {"Bella", 1, DOG}, {"Lola", 2, CAT}}; numAnimals = 3, sort = 'n'	sortedAnimals[0] = {"Bella", 1, DOG}, sortedAnimals[1] = {"Charlie", 3, CAT}, sortedAnimals[2] = {"Lola", 2, CAT}	Tests whether the function can sort the array of animals by name in ascending order. It checks whether the resulting array of animals is sorted correctly.
testSortAnimalsByAge	Animal Array = {{"Charlie", 3, CAT}, {"Bella", 1, DOG}, {"Lola", 2, CAT}}; numAnimals = 3, sort = 'a'	sortedAnimals[0] = {"Bella", 1, DOG}, sortedAnimals[1] = {"Lola", 2, CAT}, sortedAnimals[2] = {"Charlie", 3, CAT}	Tests whether the function can sort the array of animals by age in ascending order. It checks whether the resulting array of animals is sorted correctly.
testSortAnimalsByType	An array of Animal structs with 3 elements	A sorted array of Animal structs by type	This test case checks the functionality of the "sortAnimals" function, which should sort an array of Animal structs by type. The expected output is a sorted array of Animal structs by type, where each Animal struct's "type" field is in ascending order.

testCountAnimalsByTypeOneOfEachType	An array of Animal structs with one of each animal type	An array of AnimalCount structs with a count of one each	This test case checks the functionality of the "countAnimalsByType" function, which should count the number of animals of each type in an array of Animal structs. The expected output is an array of AnimalCount structs, where each AnimalCount struct's "type" field is the type of an animal in the input array and each AnimalCount struct's "count" field is 1.
testCountAnimalsByTypeMultipleOfOneType	An array of Animal structs with multiple animals of one type	An array of AnimalCount structs with the correct count of each type	This test case checks the functionality of the "countAnimalsByType" function with an array of Animal structs that contains multiple animals of the same type. The expected output is an array of AnimalCount structs, where each AnimalCount struct's "type" field is the type of an animal in the input array and each AnimalCount struct's "count" field is the correct count for that type.
testCountAnimalsByTypeEmptyArray	An empty array of Animal structs	An array of AnimalCount structs with a count of zero	This test case checks the functionality of the "countAnimalsByType" function when passed an empty array of Animal structs. The expected output is an array of AnimalCount structs, where each AnimalCount struct's "type" field is the type of an animal and each AnimalCount struct's "count" field is 0.