C PROGRAM (PART 1)

First program is written in ANSCII C and is responsible for calculating cetacean mammals locations based on provided files with observers locations and mammals sightings.

Code is split into several functions responsible for different tasks: reading files, calculating results and writing results to another file. There are also some adjuvant functions used to avoid code repetition (e.g. getting number of lines in the file) or make code more readable (e.g checking whether the mammal is located in " our sea ").

While writing into a file results have been written in reverse order to match the provided example answer, however it does not affect execution and results of the second program in any way.

Most of the data in the program is stored in arrrays of structs. There are three types of structs used: observersData (storing observers latitude, longitude and ID), sightingData (storing observers ID, type of sighted mammal, bearing and range) and results (storing calculated mammals latitude and longitude).
Although I have chosen to use arrays, I believe that using linked lists would make the program more efficient as in the current state before data can be saved program needs to read data from the files twice - first time to find number of entries in the file necessary to create an array of this size and second time to actually save data into the array. Using linked lists would allow to skip the first reading.

All variables have been given sensible names making it easy to understand what they are used for. Apart from comments explaining each functions general purpose, parameters and returned values, some additional comments have been used within the code for parts that are not self-explanatory.

To make the program work correctly files containing observers and sightings data should be either placed in the working directory of the program or the path needs to be specified. However file with calculated results will be always saved in a specific place outside the project (to make it easy to reach by second part of the program) so no path should be used.

In general program works correctly with all provided example files and displays the results in an easy to read way. Code and comments are neatly and consistently formatted, but I believe there are parts where I could have implemented more efficient solution.

C++ PROGRAM (PART 2)

Second Program is written in C++ 11 and is responsible for finding entries referring to the same mammal and identifying pods of mammals.

Program contains several functions responsible for: reading data from the file produced by first program, finding repeating mammals and finding mammals pods. Some adjuvant functions have been used to avoid code repetition and make it easy to read and understand (e.g. function checking if a mammal has already been added to the pod).
Although I have tried to make each function responsible for a single tasks, some of them still contain long pieces of code and I believe they would benefit from splitting them further into smaller functions (e.g. separating calculations and comparisons from eventual printing results)

Program reads the file from a specified location outside the project (the same as the location that first program saves into) so it only requires name of a file to read from and no path.

Code makes use of provided Location class through another class containing all mammal information - MammalData. This has been done to connect mammal location to a corresponding type as it is necessary while checking whether two of the mammals are in the same pod.

Data is mostly stored in lists of mammals (using vector from C++ Standard Library) as during the program execution data is frequently added or removed from the list (e.g when repeating mammals entries are found or while adding mammals to pods).

Code has been thoroughly commented with short descriptions, parameters and returned value of all functions as well as some further explanation of more complicated or unclear parts inside them.

This program does not save the data in any file. All results are displayed and discarded whenever user exits it.

Overall program seems to work well under all circumstances as no mistakes were found while performing single functions or whole program tests. However I believe some functions would benefit from splitting them further (so that they focus on one task only) to make the code easier to understand and update.