Please add to this list freely and write your own comments here, it will be much easier to understand and expand the code if we share what we have figured out

TO DO:

1. Assign issues(?) in gitlab to team members (BEFORE NEXT TUTORIAL)
2. Divide movement and evaluation phases into functions, describe those functions with comments (BEFORE NEXT TUTORIAL)
3. Change create\_board so that it first generates the coordinates to fill in with ones (enough to place all the penguins), and only then fills in the rest with randomly generated numbers (BEFORE NEXT TUTORIAL)
4. Expand are\_coordinates\_good so that it recognizes all the possible errors and prints info about what went wrong (similar to check\_data)
5. Combine the functions for placing the penguin – the do while loop and collect fish – in one function place\_penguins
6. Find a way of depicting the placement of different penguins on board – in my opinion we can change the value of the floe on which the penguin is standing to the decimal value of the char we use to identify the player (player A, B, C, …) and display the floes with values >=’A’ as chars, then if the board is

1 3 2 2 A

0 2 2 1 1

2 B 2 3 1

A 3 2 1 1

2 1 1 0 B

We know that player A’s penguins are in (4, 0) and (0, 4), and player B’s in (1, 3) and (4, 4).

Depending on the form of coordinates we want to interact with:

* Modify putting in coordinates so we put in x as a letter and y as a number (they’re both still stored as numbers)

OR

* Modify display\_board so that the columns are named with numbers too

Some tips:

* The code is now divided into .c files – main, setup, board, penguin, move. In those other than main you should put all the functions you write. You can later use them in main, as all the header files have been included there
* In the .h file you put the declaration of a function, and the function itself is put into the corresponding .c file
* In the structures.h file are all the structures and constants we use throughout the code. It should be included in every header file so that each function has access to them
* The board is now local to main(), so if you want to use it in a function, you need to use a pointer. In the declaration of a function you then put int board[][N] – this means the same thing as int\* board[N], but is a bit more intuitive. When you use the function this argument is just board – the type is implicit. Examples can be found in the code
* It may happen that you need to use a function from one .c file in another (for example collect\_fish will be needed both for placing and moving the penguin). To avoid redefinition research #ifndef – you can use the structures.h file as a guideline as it has been used there
* THESE CODES ONLY WORK WHEN THEY ARE A PART OF ONE PROJECT – I do not know whether when you download the git folder as a whole it automatically creates a project. If it doesn’t, you will need to do it manually. If you want to create more files, they also need to be included in the project!
* If any of you know how to create and use a makefile it would be amazing if someone did it.
* If you need to use change some data of a player inside a function – an argument of said function is a pointer “struct players\* player” you will need the operator ->. It is covered in the slides for the 6th lecture, page 15. This operator lets you access the data stored in the structure your pointer is pointing to, f.ex. if pla\* is a pointer to struct players player, in order to access player.fish / player.penguin[i].x you write   
  pla->fish / pla->penguin[i].x. An example of this is the function collect\_fish
* If you have any suggestions about the structure of the project PLEASE tell me, this is the first time I’m doing this and it’s rather confusing

As of now the setup phase works – you may run it, generate the board and place the penguins!