REPORT 3

(Program structure)

- Files (headers)
- Main functions

Headers:

- **Player.h**
  contains functions concerning players, like: creating arrays for players, increasing the score and decreasing the number of penguins
- **Board.h**
  contains functions concerning the board, like creating and printing it out
- **Placement.h**
  contains functions that will handle the placement of penguins on the board
  uses player and board arrays previously created
- **Movement.h**
  contains functions handling the movement of the penguins
  uses increasing the score and decreasing the number of penguins functions from player.h
- **File_management.h**
  will contain functions that will manage the files, like reading from the board and rewriting the board
- **Board_gen**
  this file is just for testing purpose. It creates a board as per the instructions and saves it in a text file.

Functions in:

Main

- **createPlayers** is used
  (player.h)
- **makeBoard** is used
  (board.h)
- **printBoard** is used
  (board.h)
- **showWinner**
  compares the scores of each player, stored in playerList, and displays the winner

Player.h

- **createPlayers**
  (from player.h)
  Given the number of players it creates an array playerList[nr of players][3] which will store information about players

```
playerList[i][0] = i; //playerID
playerList[i][1] = penguinsAvailable;
playerList[i][2] = score;
```

- **increaseScore**
  given the information from board array (number of fish on the tile which the penguin is standing on) increases the player's score in playerList array

- **decreasePenguins**
  it decreases playerList[i][1] - the number of available penguins, when each tile around the penguin is non active or occupied, so that the penguin cannot move anywhere

Board.h

- **makeBoard**
  a function that given the size of the board creates a 3d array board[m][n][4]

```
//board[row][col][0] - numFish (1 to 3, number of fish on the tile)
//board[row][col][1] - isPenguin (0 if there's no penguin on the tile 1 if there is)
 //board[row][col][2] - active (whether it's active or not, 1 is it is, 0 if not)
 //board[row][col][3] - playerID (id of the player on the tile, 0 if there's no one)
```

  intitialy:

```
board[row][col][0] = rand() % 3 + 1;
board[row][col][1] = 0;
board[row][col][2] = 1;
board[row][col][3] = 0;
```

- **printBoard**
  (used in main)
  a function that prints out the board of given size (number of columns and rows)

Placement.h

function	description
int **placing**(int number_of_players, int penguins, int m, int n, int board[m][n][4])	set of functions responsible for interactive mode placement phase (in some way an equivalence of main in placing_penguins.h) int penguins is a number of penguins that are to be placed by each player
int **is_board_correct**(int m, int n, int board[m][n][4])	checks if board was generated correctly, returns 0 if correct, otherwise 2/3 in case of an error

int **penguins_left**(int m, int n, int board[m][n][4], int penguins, int player_number)	calculates how many penguins are there left to place, returns the amount of penguins left to place and stores it in penguins variable player_number is ID of the player that is currently placing his penguin
int **tiles_with_one_fish**(int m, int n, int board[m][n][4], int total_penguins_to_be_placed)	calculates number of tiles with one fish on it - tiles on which penguins can be placed
printBoard is used	function from board.h prints updated board on the screen
void **place_penguin_on_tile**(int m, int n, int board[m][n][4], int *pointer_to_board, int player_number)	asks user for coordinates (m,n) of the tile which they want to place their penguin (if they are invalid then it keeps asking until the given coords are valid), overrides an array with new data uses **printBoard** to print out the current state of the board
int **save_to_file**(int m, int n, int board[m][n][4])	saves board into the board.txt file, returns 0 if done successfully, in case of an error returns 2/3
int **placing_automatic**	(to be done) will be responsible for automatic mode placement phase

(flowchart below)

Movement.h

- int **play**(number of players, board, playerList)
  the main algorithm that is used to play the game. It utilizes phaseIsOver(), moveOver(), isValidMove(), makeMove();

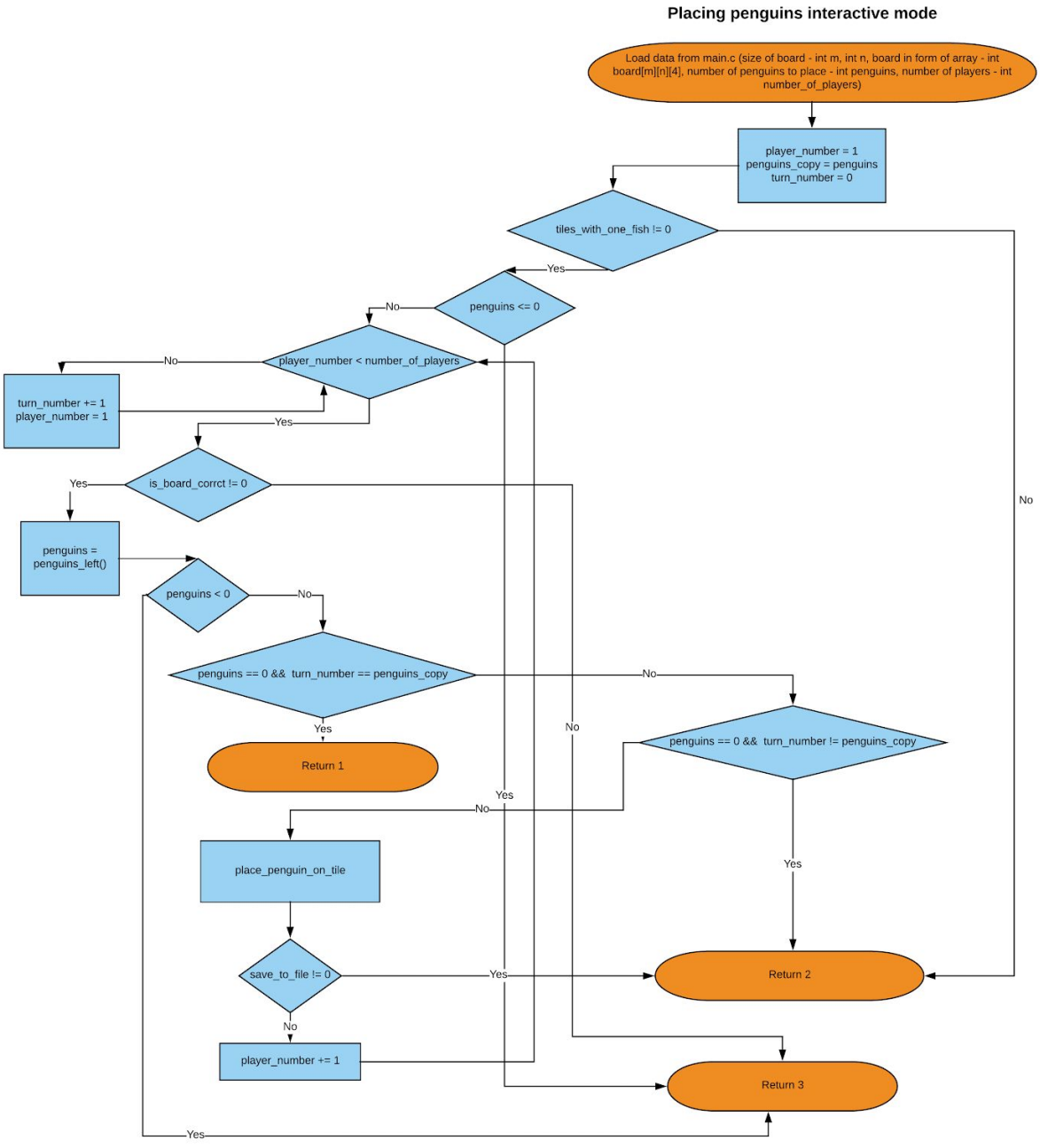
- int **phaseIsOver**(movesLeft)
  returns 1 if the phase is over and 0 otherwise. It checks the size of movesLeft. If sizeof(movesLeft) == 0, it means that there are no more moves left.

- int **isValidMove**(move, board, playerID)
  returns 1 if the entered move is valid for a given player, 0 otherwise.

- void **makeMove**(move,board, playerID)It applies the move to the board if isValidMove(move, board, playerID) returns 1.

File_management (in the phase of planning)
- function that reads from the board
- function that rewrites the board

Placing.h flowchart



board_gen.c flowchart: