Division of tasks:

1. Add GameState Structure with all the variables for the game:
Patryk

2. Change Variables inside the functions to GameState Structure:
Jasmine

3. Dynamic 2D Board Generation:
Chris and Aman

4. Printing out the Board:
Shaunak Bhardwaj

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include "userinteraction.c"
#include "generate.c"
#include "placement.c"
#include "movement.c"
#include "printing.c"
#include "game_struct.h"  // Header file for file where we store all our structures

int main()
{
    GameState Game; // Creating a Game variable that will hold informations needed to run the program
    welcome(); // User interaction - showing communicate at the start of the program
    printf("Hello again! Please enter the number of rows and columns \n");
    scanf("%d", &Game.board_width); scanf("%d", &Game.board_height);
    // User enters values to the Game structure
    // User Interaction - Telling user to input rows and columns
    Field *board[Game.board_height];
    // Creating a board for our game using Field structure
    generate(Game, board);  // Changed variables we use to Game structure and board

    print_board(Game, board);

    placePenguin(Game, board);

    move_penguin(Game, board);

    ScoreBoard(Game);

    return 0;
}
```

```c
#ifndef STRUCT_H_
#define STRUCT_H_

typedef struct  // Struct for our board
{
    int fish_no;
    int player_no;
}Field;


typedef struct Game   // Struct containing all of the variables that the program needs.
{
    int board_height;   // The height of the board.
    int board_width;    // The width of the board.
    int coordinate_1;   // Coordinate of penguin's location.
    int coordinate_2;   // Coordinate of penguin's location.
    int coordinate_3;   // Coordinate of where we want to put the penguin.
    int coordinate_4;   // Coordinate of where we want to put the penguin.
    int player_count;   // Number of players.
    int total_players;  // Total number of players.
    int fish_count; // Number of fishes
    int score;  // Score.
    int player; // Player.
    int round_number;   // The current round number.
    int initial_fish_number; // Number of fish at the start of the round.
    int current_fish_count; // Current number of fish after the round ended.
    int starting_score; // Score at the start of the round.
    int board_score[20];    // Board score.
    char player_name[20];   // Name of the player.

}GameState;

#endif
```

```c
#include "game_struct.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>

void generate(GameState Game, Field *board[Game.board_height])
{
    int i, j;
    //This for loop assigns pointers from our array of pointer
    // to the beginning of memory blocks with size of Field struct
    // There are board_width number of smaller memory blocks
    // And board_height number of bigger memory blocks
    for (i = 0; i < Game.board_height; i++)
    {
        board[i] = (Field *)malloc(Game.board_width * sizeof(Field));
    //malloc assigns a block of memory and returns the pointer to the
    //beginning of this block
    }

    for(i = 0; i < Game.board_height; i++)
    {
        for(j = 0; j < Game.board_width; j++)
        {
            board[i][j].fish_no = rand() % 3 + 1;
            //upside of this method
            // is that it does not change the way we refer to this array
            // later on in the program
            board[i][j].player_no = 0;
        }
    }
    return;
}
```

```c
#include "game_struct.h"
// In order to check if the move is correct we need our board, in order to have
// our board we need it's height and width, that's first 3 variables inside the function
// then we need coordinates of where the penguin currently is (coordinate_1 and coordinate_2)
// and coordinates of where we want to put the penguin (coordinate_3 and coordinate_4)

int checkMove(GameState Game, Field *board[Game.board_height])
{
    return 1;
}
```

```c
#include "game_struct.h"
// Phase == 1 : Placement

/* In this phase we give each player a chance to put his penguin on the field
for loop executed as total amount of penguins
Inside of it a do while loop that will be executed if the player move is wrong
outside of while loop we change player to the next one */

// Defining var for placement phase

void placePenguin(GameState Game, Field *board[Game.board_height]) {

    return;
}

/* How the placement phase function is gonna work
void placePenguin ( Penguin *penguin , Field *field )  {
        penguin -> field = field;

    for ( int i = 0; i < player_count; ++i) {
        for ( int j = 0; j < players[i] -> penguinsCount; ++j ) {
            if ( players[i] -> penguins[j] == penguin ){
                players[i] -> score += penguin-> field -> fish_count;
                penguin -> field -> fish_count = 0;
            break;
            }
        }
    }
} */
```

```c
#include "game_struct.h"

void print_board(GameState Game, Field *board[Game.board_height])
{
    for(int i = 0; i < Game.board_height; i++)
    {
        for(int j = 0; j < Game.board_width; j++)
        {
            printf("%d%d ", board[i][j].fish_no, board[i][j].player_no);
        }
        printf("\n");
    }
}
```

```c
#include "game_struct.h"

void welcome()
{
    printf("Hey there! Lets play ''HEY THAT'S MY FISH'' \n");
}

/* When the movement phase will be finished (There no more moves))
Check who won and who lost
Print out the score and winner
printf("Game over! %s you won, %s you suck", winner, loser)
END */
//  Each round the function will store the the starting score of the player
//  The score will be the difference between the initial amount of fish before the start of the round and the current number after it ended
//  So at the end of the round the current number of fish becomes the initial number
//  At the same time the starting score becomes the sum of previous starting score and the score of the round


int ScoreBoard(GameState Game)
{
    return 1;
}
```

```c
#include "game_struct.h"
// Phase == 2 : Movement
/* In this phase we give each player a chance to move his penguin
    1. Inside of it we will have a do while loop that will execute until there would be no more moves
    2. First player chooses the penguin (For user interaction we print out his name and tell him to choose a penguin)
    printf("%s choose a penguin", player_name[player_number][];
    scanf("%d %d", &coordinate_1, &coordinate_2) coordinate_1 - coordinate one, coordinate_2 - coordinate two
    Now we check if the penguin he has chosen has a valid move
    3. Placing the penguin (For user interaction we print out his name and tell him to place a penguin)
    printf("%s choose a penguin", player_name[player_number][];
    scanf("%d %d", &coordinate_3, &coordinate_4) coordinate_3 - coordinate three, coordinate_4 - coordinate four
    Now we check if the move is valid
    The move is made
    We update the scoreboard */
void move_penguin(GameState Game, Field *board[Game.board_height])
{
    // The main movement phase will happen here
    // We will loop through each player checking if they have a valid move and if they have one letting them choose which penguin to move.
    // Then updating the board and the score of the player. Then continue to loop through the players till no possible moves are available
    // checkMove(board_width, board_height, Field board[board_height][board_width], coordinate_1, coordinate_2, coordinate_3, coordinate_4)
    return;
}
```