Team 7

Report 2, Meeting 3

Data structure choice

Array is the data structure of choice for this project.
As mentioned in our previous report, due to the difference in programming experience among members, we tend to go with the simplest and the most efficient solution, as it helps us effectively and easily access data for each player.

We rejected:
-*stacks and queues* - because neither first-in-first-out nor first-in-last-out structures can be effectively used without complicating the algorithm;
-*Linked lists* - because we're not working with huge amounts of data (we would achieve only insignificant amount of runtime decrease but would over complicate the coding process);
-We thought about using *binary search trees* at the endgame phase to keep track of the score but, again, the result would be spaghetti code.

Interaction with user

Inside the loop we keep asking users for input, which (for now) are the coordinates of the tile they want to move their penguin to.
Then the algorithm checks if this move is valid.
If it is, it applies the move and the board updates along with all the player values (scores, penguins left, etc).
If the entered move is not valid, it prints an error and asks the user to enter valid coordinates.

The other idea is to make is the user-interaction part arrow-button based instead of coordinate-based for more intuitive use, but we'll apply it only if we have enough time.

Main Loop Idea

To store the data for each player we'll use headers and arrays. To help us keep track of things, we break different aspects of the input .txt file apart into smaller arrays.
(For example, we record the values associated with each tile in separate cells, that correspond to that tile, of the array *board.)*

Then our algorithm checks which phase is the current one -- *placement* or *interactive.*
Then we start a loop that corresponds to the phase.

Inside these loops, we update the information stored in arrays, we update the *board* array and update the input file after each move.

Once there are no more moves left, the loop breaks and the winner of the game is showed.

```
#include <stdio.h>
#include <string.h>
#include "board.h"
#include "player.h"

int main()
{
	// insert code here...
	int m = 3, n = 4;
	int board[m][n][4];
	int numberOfPlayers = 2;
	int playerList[numberOfPlayers][3];
	char phase[] = "placement";

	createPlayers(numberOfPlayers, playerList, 3);
	makeBoard(m, n, board);
```

```
	printBoard(m, n, board);

	if (strcmp(phase, "placement") != 0) // if gamephase == placing, do placing until it's over
	{
		// do placing
	}
	else if (strcmp(phase, "movement") != 0) // if gamephase == movement, do movement until it's over
	{
		// do movement
		while (phaseIsOver() != 1) {
			// play();
		}
	}
	// print scores, determine winner
	return 0;
}
```