CS3210 – Parallel Computing (AY 2017/2018 Sem 1)

Assignment 2

MPI Football (15 marks)

Individual Submission due on **Fri, 17 Nov, 2pm**

In this assignment, you must simulate a game of football in MPI, where **players from two teams** compete on a **football field** (also called football pitch). You must implement this game using MPI programming. You are required to implement each player as a separate MPI process and the field as a set of MPI processes.

The football field size is 64 meters in width and 128 meters in length for the training session and 96 meters in width and 128 meters in length for the match. You must consider that the field as a grid, with the grid element of one meter by one meter. The players move from one grid element to another in increments of one meter. The player can move in four directions on the field: top, right, bottom or left. There are two goal posts on the pitch, located on the extreme ends, at coordinates (0, 43) to (0, 51) and at coordinates (127, 43) to (127, 51).

Football players cannot be trusted with too much information (they easily lose focus). Therefore, a player works as follows: whenever he needs to find out the position of the ball or the position of another player, he must inquire the field for this information. **The player is not allowed to guess or to know beforehand the position of the ball or of another player.** This is a strict requirement in this assignment. However, the player knows his own position.

The assignment is structured in two parts. The first part asks you to simulate a football training session by one team, on a field. The second part asks you to simulate a game between two teams.

Football Training Session (5 marks)

You are required to create an MPI program that simulates a football training session. During this training there is only one team on the field, and the players of the team pass the ball around to warm-up and practice their shots for the big upcoming match.

Your program must be structured in the following way:

- There must be 11 player processes. The player processes only know information about themselves. If they require any information about other players or about the ball, they must retrieve it from the field process. **Player processes are not allowed to exchange data with other player processes.**
- There is one field process. When inquired, this process will disseminate to player processes information about the position of the ball or of other players.

The training session consists of 900 rounds. One round proceeds as follows:

1. All players determine the grid element where the ball is located. The ball is assumed to be stationary.
2. All players move in the direction of the ball, and they stop when they have reached the ball or when they have ran 10 meters:
   A. If they reach the grid element with the ball:
      1) If they are alone on the grid, they win the ball.
      2) Else, one of the players randomly wins the ball.
      3) The winner of the ball kicks the ball to a random location on the grid. The winner informs the field of the new ball location.
      4) They remain on the grid element where they stopped running.
   B. If they do not reach the ball in 10 meters, they stop after running 10 meters and remain on that grid element.
   C. All players finalize their run for each round before you can decide which one wins the ball.
3. The players send their new location to the process of the field in charge of their location.

Each player must collect the following pieces of information about his own training:
1. How many meters he has run.
2. How many times he has reached the ball.
3. How many times he has kicked the ball.

After each round, the field process must output to the console the following information:
1. Number of the round
2. Coordinates of the ball, with dimension on long side of the field first.
3. For each player, starting from id 0 to id 10: Id, initial coordinates, final coordinates, whether they reached the ball, whether they kicked the ball, meters run so far, times he reached the ball, times he kicked the ball

Example:
…
15 // This is round 15
109 25 // Ball was at coordinates (109, 25). Coordinates are given as long side of pitch first.
0 106 21 109 25 1 0 78 2 1 // Player 0: initial position (106, 21) final position (109, 25), 1 = he has reached the ball this round, 0 = he did not win the ball this round, 78 = he ran 78 meters from the beginning of the training, 2 = he reached the ball twice since the beginning of the training, 1 = he won the ball once from the beginning of the training.

For the football training session, you need to use process-to-process communication. You may choose blocking or non-blocking communication. You cannot use collective communication.

Football Match (10 marks)

You are asked to create a MPI program to simulate a football match between two teams.
For the football match, you must simulate two teams of player (Team A and Team B) and one field. This time, the field consists of 12 processes, where each process is in charge of a patch of 32 meters by 32 meters from the field. Field process 0 (FP0) will be in charge of the part between (0,0) and (31,31) and so on. The assignment of field grid elements to MPI processes is shown in Figure 1. You must respect this assignment in your implementation.

0, 0→31 ↓ FP0 31	0, 32→63 ↓ FP1 31	0, 64→95 ↓ FP2 31	0, 96→127 ↓ FP3 31
32, 0→31 ↓ FP4 63	32, 32→63 ↓ FP5 63	32, 64→95 ↓ FP6 63	32, 96→127 ↓ FP7 63
64, 0→31 ↓ FP8 95	64, 32→63 ↓ FP9 95	64, 64→95 ↓ FP10 95	64, 96→127 ↓ FP11 95

Figure 1: Assignment of field positions to MPI field processes

The players have a more complex behavior compared with the training session. Each player has three attributes: **speed**, **dribbling skill** and **kick power**. The speed attribute dictates the maximum distance the player can run in one round. The dribbling skill influences the likelihood that the player wins a challenge for the ball. The kick power dictates the maximum distance the ball can travel, if kicked by the player. All attributes have values between 1 and 10. For fairness, each player has speed + dribbling + kick = 15. That means a player can be very good at one attribute and bad at the others, or mediocre at all three.

A football game is made up two halves, each consisting of 2700 rounds. In the first half, Team A guards the left post and attempts to score at the right post. In the second half it is the other way around.

A round proceeds as follows:

1. Players determine the grid element that holds the ball. The ball is assumed to be stationary.
2. You can choose which players run in the direction of the ball and how much, subject to these two restrictions:
   - A player cannot run more meters than his speed skill
   - A player cannot run more than 10 meters, irrespective of his speed skill.

   A. If they reach the grid element where the ball is:
      1) If they are alone on the grid element, they win the ball.
      2) Else:
         i. They pick a random number between 1 and 10.
         ii. They multiply this number with the **dribbling skill**. We call this product **the ball challenge.**
         iii. They send **the ball challenge** to the field process on which the ball is located.
         iv. The field process on which the ball is located receives **the ball challenges** from all processes challenging the ball. The field process chooses the winner

as the player with the highest **ball challenge.** If two players have the same **ball challenge**, the field process randomly chooses one of them.

3) The winner of the ball determines a location on the field where to send the ball. This location must be at most **kick power * 2 meters.** The winner kicks the ball to that location. (There is no offside rule, therefore a player can chose any direction for his kick.)
4) They remain on the grid element.

B. If they do not reach the ball in 10 meters, they stop after running 10 meters and remain on that grid element.

3. You can choose that some player DO NOT run towards the ball. This entails that you are free to move these players anywhere on the field, but cannot move a player more meters than his speed skill (the 10 meters does not apply).
4. The players send their new location to the process of the field in charge of their location.

Special cases:

1. If the ball is kicked left of coordinates (0, 43) to (0, 51) or right of coordinates (127, 43) to (127, 51) the team has scored a goal. The ball is restarted from middle of the field (64, 48).
2. You are free to decide what to do in case the ball is kicked out of the field.

You must implement the following features:

1. All communication must be done using collective communication.
2. You must have exactly 34 processes (22 players + 12 field processes).
3. Each team is part of a communicator. You are free to communicate anything between players of the same team, with the exception of the positions of the ball or of other players. The players may chose the communicate strategies, instructions etc. Players can pass the ball around.
4. Between rounds a player must not remember anything related to other players. This means that if one player in a team wants to know the attributes of another player, it must use that information during that round.
5. Players from different teams must not exchange data.
6. After each round, field process 0 must output to the console the following:
   a. Number of the round
   b. Coordinates of the ball, with dimension on long side of the field first.
   c. For each player, starting from id 0 to id 10 of team A: Id, initial coordinates, final coordinates, whether they reached the ball, whether they kicked the ball, ball challenge. (The challenge is -1 if they have not reached the ball). After team A, same for team B.
7. You are free to choose the player attributes' values, subject to the restriction speed + dribbling + kick = 15. You can choose the player starting location in the match. The initial starting location of the ball is the center of the pitch.
8. You must program both teams to be competitive – i.e. an attacking team should attempt to score goals and a defending team should try and prevent the other team to score. **Passing the ball randomly does not count as competitiveness.**
9. You can also program Team A and Team B to play by different strategies. You are free to choose the initial position of the players. You have a lot of processing power that you can harness – be creative.

Deliverables and Submission

There are two target machines for this assignment:

a) [15 marks] the 3-node cluster in the lab

b) [2 bonus marks] the NSCC supercomputer: https://user.nscc.sg/saml/

Prepare a **zip file** using your **student id (matric no), e.g.** A01234567X.zip, which contains:

training_mpi.c match_mpi.c	You need to provide two source code files, one for the training session and one for the football match. The source code must be properly commented and indented.
makefile.lab	For compiling your programs on the lab machine.
runfile.lab	The command to run your programs. We will execute this command from node 1 (intel i7).
machinefile.lab rankfile.lab	Additional configuration file that you need for the execution in the lab.
training.lab.o match.lab.o	A sample of the output of each programs (you need to redirect the output and error to a file)
makefile.nscc	For compiling your programs on NSCC supercomputer.
match.pbs	PBS job script for submitting your match_mpi.c program to NSCC supercomputer.
match.pbs.o match.pbs.e	Summary reports from your execution on the NSCC supercomputer.
assignment2_ report.pdf	Contains: 1. Pseudo-code of your solution for both parts. 2. A diagram of your communication between the MPI processes, within a game round. 3. A walkthrough of your design, highlighting the important design decisions. 4. Instructions on how to build and run your program, as well as the configuration (machinefile or rankfile) with which you run the program.

This assignment is to be done on an individual basis. You can discuss the assignment with others as necessary but in the case of plagiarism both parties will be severely penalized.

Score breakdown:

Design [8 marks]	Evaluation of your approach on partitioning, communication, and mapping. We look for well thought out design with **strong rationale.**
Implementation [4 marks]	Code quality. We look for modularization, documentation and good coding style. (extra 1 bonus mark allocated for NSCC setup)
Result Analysis [3 marks]	The execution timing and analysis on the lab machines. (extra 1 bonus mark for NSCC supercomputer analysis and comparison). We look for careful collection of results and thoughtful analysis of the result.

The zip archive must be uploaded to IVLE in the workbin folder "Assignment2" by **Fri, 17 Nov, 2pm. Penalty of 10% per day for late submissions will be applied.**