COMS20001 Concurrent Computing (2018-19)

# Assignment Weeks 3-4

(pair programming work, purely formative, 0%)

# xCore-200 Explorer Kit:

## Console Ant Defender Game

**Your Task: xCore-200 Ant Defender Game**
*(To work on this assignment please pick up your team's board during the labs on 19/10/18.)*

**Objective.** Work in your teams of two for this assignment. This assignment aims at introducing you to the basic I/O of your XMOS hardware and bringing it together with parallel execution and channel communication.

You are given an XC code skeleton that provides you with some first program structure and helper routines towards implementing a very rudimentary console movement game that utilises your xCore-200 Explorer board for button input as well as LED feedback.

**Game Idea.** Your task is to extend the given skeleton code to implement the following game: A left-to-right, 1-dimensional string world of 23 character locations (index 0 to 22) contains two types of territories: "treasure" represented by character "**-**" in locations 8 to 14 and "desert" represented by character "**.**" in all other locations.

The world is circular, going to the right at location 22 leads back to location 0 and vice versa. Two ants roam this world: a defender ant (defending the treasure) represented by character "**X**" and controlled by the user, and an attacker ant represented by character "**O**". During the game, the user has to defend the "treasure" locations by blocking attacks from a (program-controlled) attacker ant. The user controls the defender ant using two buttons on the xCore-200 Explorer board. The world is frequently printed to the console and LEDs on the board indicate the game status.

**Defender Ant Details:** The user controls the defender ant by pressing either button SW1 (moving 1 left) or button SW2 (moving 1 position to the right). The defender ant can only move to a position that is not already occupied by the attacker ant. A particular **controller** process which has information about both ants either grants or denies move attempts. The defender's starting position is 11.

**Attacker Ant Details:** A second ant is controlled by the system and starts at position 2. It attempts moving in one direction (either right or left). This attempt is denied if the defender ant is already located there, in this case the attacker ant does not move for its current attempt, but changes direction for its next move.

To make the game more interesting: before attempting the n-th move, the attacker ant will change direction if n is divisible by 31 or 37. The game ends when the attacker ant has reached any one of the treasure locations.

**About the Skeleton Code:** Your task is to implement further code in the processes **userAnt**, **attackerAnt** and **controller** according to the game description. The defender is controlled by a process called **userAnt**, which has channels to **buttonListener**, **visualiser** and **controller**. The process **buttonListener** is implemented for you; it sends the button inputs to **userAnt**, the values 13 and 14 sent along this channel indicate either of the two buttons being pressed, respectively. The **visualiser** is also implemented for you; it waits to receive the current position from the ants and instructs a printing routine to print the world on the console. It also controls LEDs on the board, alternating the status of a green LED between moves, activating the blue LED whenever the attacker ant is blocked, and switching on the red LED if the attacker ant approaches the treasure. If the attacker has reached the treasure all LEDs are activated. The attacker ant is controlled by a process **attackerAnt**, which has channels to the **visualiser** and **controller**. The **controller** process responds to ''permission-to-move'' requests from **attackerAnt** and **userAnt**. The process also checks if the **attackerAnt** has moved to any treasure location and terminates the game in this case stating to the console that the defender lost...

Once you have implemented a working, stable and well playable game and have time left, extend the code skeleton so that **all** threads are terminating after the game has ended (graceful shutdown).

**Opportunity to Present your Game:** This task runs over two weeks (4h of lab time) and is formative. It prepares you for the marked coursework of TB1 in the unit. In any case, your game should at least compile and run, and it should implement the game mechanics without deadlock. Make sure you structure and comment your code well. Develop your program in small steps to try and avoid complicated bugs.