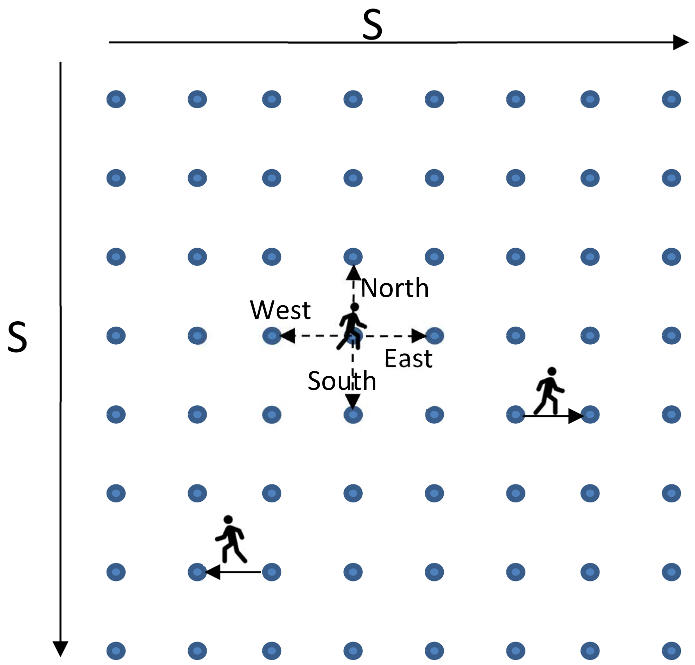
CSCM98 Coursework
The Sunday Walker

Deadline: 19th of November 2024, 11am

Coursework Submission:

To work on this assignment, you will have to download the visual studio solution available on the course website. Submission will be done on canvas by uploading the *main.cpp* file. C++ comments can be added to the file if needed. All code and work should be submitted as a single .cpp file so that testing can be done uniformly. Please follow the submission guidelines carefully as deviating from them will result in a loss of marks. Note that the work submitted must be your own work and you should not help or receive help from others. If you use extra material from somewhere else, please reference it by adding comments in the code.



Description

N walkers (represented by *N* threads) are located on a 2D square grid of size *S*×*S*, with both an initial position and a final destination calculated before starting the game. Walkers will move randomly along edges of the grid until they reach their final destination, an edge being a straight horizontal or vertical line between two neighbouring positions. Therefore, walkers can only be located on either a vertex or on an edge of the grid. Once on their final destinations, walkers will be removed from the grid but the number of walkers reaching a particular position will have to be recorded. There are a few limitations though that will need to be considered:

- Walkers can only move along one edge at a time along a North, South, East or West direction.
  - The *WalkerI* thread needs to implement a while loop running until its final destination is reached.
- Direction is picked randomly at each step (See *RandomWalk* function).

- Only 3 active walkers can be at the very same location at the same time. There should be a minimum length of time spent at a specific location. This is implemented with a constant in the code and a call to a *sleep*_for function (See *WaitAtLocation* function).
- Streets are narrow and therefore only 4 walkers can travel along a given edge at the same time. There should be a minimum length of time spent on a specific edge/road. This should be implemented with a constant in the code and a call to a *sleep*_for function (See *CrossTheStreet* function). .
- All concurrency aspects must be implemented with the two mutex functions given two you in the code. No other concurrency libraries or functions can be used.

Part of the code is already available to you and come with some definitions and an initial setup. You will have to complete the following tasks to make it compatible with the given rules, but do not try to change the parts already implemented and try to understand code first.

Marking Scheme

Thread Creation and Waiting (20 marks)

20 marks for the proper creation of threads as well as for the termination.

Coherent Grid Result (10 marks)

Marks will be granted if walkers have arrived at their destination and the result is coherent with what is to be expected.

Correctness of the Mutual Exclusion Implementation (20 marks)

These marks are for the correctness of your concurrency implementation. Data should stay coherent at all times and no deadlock should occur. We also impose the use of the two *Lock* and *Unlock* functions given and nothing else, but you can write your own extra code from these two functions. As sometimes it can be difficult to prove correctness of concurrency, submitted code will have to be as logical and coherent as possible.

Efficient traversal of the grid (45 marks)

Concurrency must be efficient, meaning that most operations can be handled in parallel when needed (15 marks). All edges must be traversed by at most four walkers at the very same time (15 marks) and three walkers at most can be at the same location (15 marks). Code should make sure these conditions are met.

Code can be tested and submission guidelines followed (5 marks)

5 marks will be given if the submission follows guidelines cited above and does not involve extra work (e.g. correcting code before compilation, code can be compiled,..).

Notes:

- Use the file given. Most code should be written in the dedicated areas.

- Any partial solution may return only partial marks. The exact marks will depend on the quality of the answers and can only be evaluated upon reception of the coursework.

    - Specifications given need to be met as closely as possible.

- This is a strictly personal work. Any external help must be referenced as comments in the code.
- Add your student ID and Name to the code.
- By submitting coursework, electronically and/or hardcopy, you state that you fully understand and are complying with the University's policy on Academic Integrity and Academic Misconduct. The policy can be found at https://myuni.swansea.ac.uk/academic-life/academic-misconduct. The consequences of committing academic misconduct can be extremely serious and may have a profound effect on your results and/or further progression. The penalties range from a written reprimand to cancellation of all of your marks and withdrawal from the University.