Interpreted requirements of the program

1. At startup, the program should read in the 2D maze configuration (“mazedata.txt”) which stores the information about the maze dimensions, barriers, start and end locations.
2. The program should have **at least** 2 threads, each thread attempting to explore surrounding locations to discover whether it contains a barrier (‘#’) or danger (‘X’).
3. The program should maintain a global Maze resource or variable, holding information about all the barriers or danger areas uncovered by your exploring threads!
4. When a particular thread has encountered a barrier (‘#’) or danger (‘X’), it should …
   1. Record the path (history of point locations) it has traversed, since the Start Location, to reach the barrier / danger areas, and locations of barrier / danger should be marked on your ‘global Maze resource’
   2. Record the path (history of point locations) it has traversed, since the Start Location, to reach the barrier / danger areas, and locations of barrier / danger should be marked on your ‘global Maze resource’
5. Whenever a thread is destroyed, the program should create another replacement thread, to traverse the jungle maze beginning from the Start Location again. But this time, it should access the ‘global Maze resource’ to learn and avoid the barriers and danger areas discovered by its predecessor threads.
6. Knowledge (of locations of the barriers / danger areas) of the destroyed threads must be recorded in the ‘global Maze resource’ that can be accessed by future generations of created threads.
7. Access to the ‘global Maze resource’ should be protected via usage of mutex locks when
   1. Updating its discovery of the path to barrier / danger areas
   2. Accessing the ‘global Maze resource’ to learn about the discovered locations of the barriers / danger areas

Only 1 thread can access it at any one time.

Diagram / Illustrations of program design

Start

Summary of Implementation

* When the program starts, it reads in and stores information of a 2D maze from “mazedata.txt”
* Displays the maze to traverse and its length, breadth, start location and end location
* Displays the unexplored maze and its length, breadth, start location and end location
* Creates a “summary thread” that waits for a signal. This thread will display as well as store the solution in a text file upon waking up (receiving a signal).
* Creates a thread that traverse the maze using the 1st algorithm
* There are 3 depth-first traversing algorithms in my design
  + 1st one goes in order North South East West
  + 2nd one goes in order South West East North
  + 3rd one goes in order West East North South
* When the 1st thread reaches a dead end, 2 other threads are spawned using the 2nd and 3rd algorithm
* So now there are altogether 3 threads running concurrently each with a different traversing algorithm
* Any barriers, danger areas or dead ends encountered by each thread is submitted and stored in the global maze resources
* If a thread reaches a dead end, it backtracks to a junction where another route is available and takes the route
* If a thread reaches a danger area
  + A new thread is spawned using the same traversing algorithm
  + The new thread starts from the beginning
  + The old thread dies
* There will be an elapse time update each time a thread reaches a dead end or hits a danger area showing
  + Number of dead end paths found
  + Barriers discovered
  + Danger areas discovered
* Hence there will always be 3 threads traversing the maze at each time using different traversing algorithm (unless the first thread finds the end location without reaching a dead end first)
* Once a thread finds the end location, it submits the solution then sends a signal
* The signal then wakes up the summary thread
* The summary thread receives the signal and prints the solution and stores the solution to a text file
* The main function waits for the summary thread to finish before exiting.

Reflections on program development

**Difficulties face**

Initially I thought threading could be a fairly had concept to grasp but after learning and experimenting it, I gained quite a good understanding of it in a short period of time. The next was to ensure only one thread access the global maze resources. After learning to use mutex, this problem was easily solved. Then I thought about the many threads that are running concurrently, and how to prevent the main program from ending prematurely or which thread should it wait for before ending the program. Hence I implemented a final thread that runs before the traversing threads waiting for a signal from the traversing threads once a solution is found. The main program then waits for the final thread to display and save the solution to a text file before exiting.

**What could have been done better / possible enhancements in future**

I could have implemented more than 3 active threads running at each moment, this way more area of the maze will be covered within a period of time making if more efficient. Also I could have explored different traversing algorithms which might be better than the current ones. I could also modify the programs such that it is able to find the shortest path better still, all available paths to the end location.

**What have I learnt**

I have learnt how to apply the concept of threading into practical programming. I also learnt the importance of mutex and how to use it. And I have learnt the usefulness of a condition variable and how to apply it.