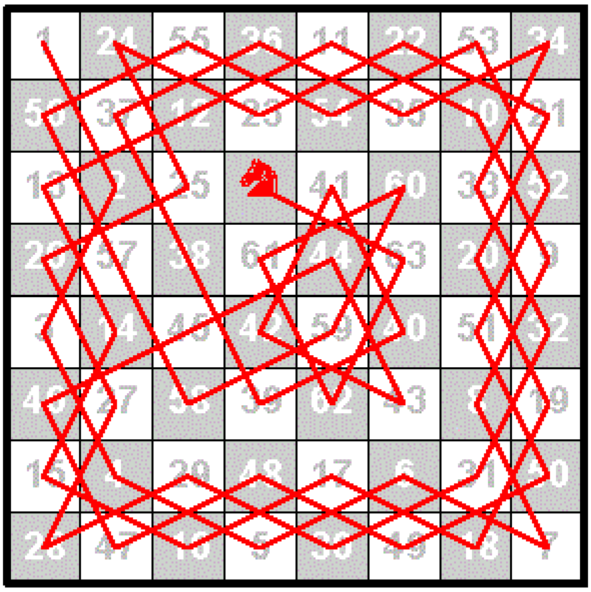
Assignment 3

Due Date

This assignment is to be completed in pairs. The assignment is due at 3PM Wednesday May 26th 2021 and should be completed with a partner. You and your partner should work together on all of the design and programming. It should be done using the *pair-programming* methodology and not by division of labour. Use the “Assignment 3 Pairs” groups on MyLO to find and register your pair.

Context

The Knight’s Tour is a challenge to move a knight from the board game of Chess around the board landing on all squares but reaching each square only once. Knights move in an ‘L’ shape (of two squares in one direction and one square at right angles to it or vice-versa).



Task

Your task is to implement the *Knight’s Tour* problem in C so that the computer will find the solution for you.

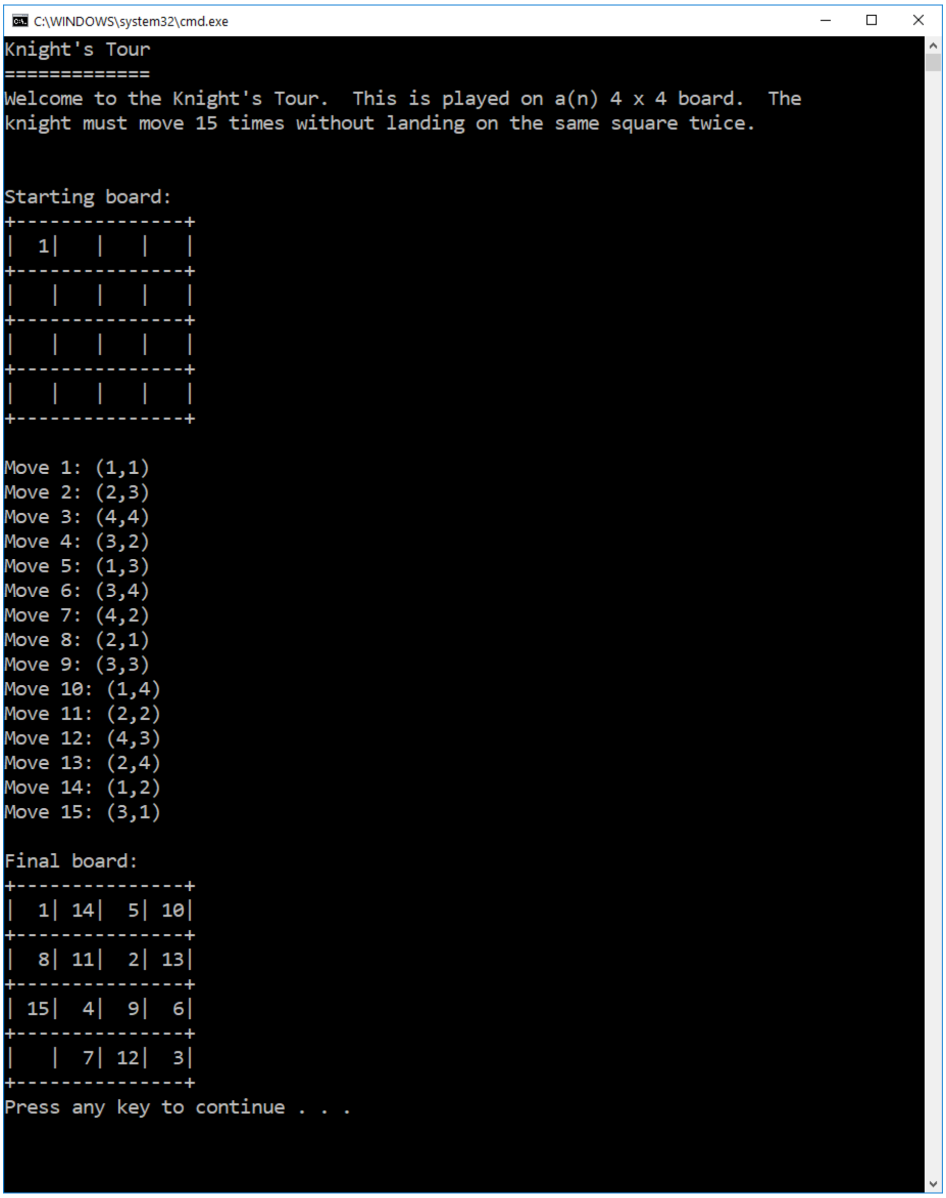
The computer will select moves by creating a *game tree*. A game tree consists of all the possible states of the game and in this assignment the computer determines any possible solution to be the best one. Each node of the game tree has children that indicate the states of the game that follow from the state of the parent for each possible move, i.e. each node in the tree possesses at most 8 children of any node but as more moves are made, or if a smaller board is used, then there will become less.

This assignment uses many data structures (trees, linked-lists, stacks, and queues) to solve the puzzle. You should create and traverse the game tree using a stack and queue as intermediate data structures. When a solution is found the solution should be displayed. The program will be text only.

A *Visual Studio* project folder is available for download from MyLO. The project contains many header (`.h`) and source (`.c`) files. All required files are present. *You should not add any further files or change any code which is provided (other than the settings in `assig_three121.h`).* You don't have to use Visual Studio if you don't wish to; any editing and compiling environment is fine.

You need to complete the functions within the program files which have been declared but for which the function bodies are missing and to update the header comments of the source files that you complete with your names, student IDs and ratio of effort.

Please note: there is not enough memory granted by the operating system to the running program to solve the depth-first problem on an 8x8 board or larger, or the breadth-first problem on a 6x6 board or larger. A sample run of the game is shown below on a 4x4 board. Please also note that there is no solution available for a tour of length 16 on a 4x4 board.



```
Knight's Tour
=============
Welcome to the Knight's Tour.  This is played on a(n) 4 x 4 board.  The
knight must move 15 times without landing on the same square twice.


Starting board:
+---------------+
|  1|   |   |   |
+---------------+
|   |   |   |   |
+---------------+
|   |   |   |   |
+---------------+
|   |   |   |   |
+---------------+


Move 1: (1,1)
Move 2: (2,3)
Move 3: (4,4)
Move 4: (3,2)
Move 5: (1,3)
Move 6: (3,4)
Move 7: (4,2)
Move 8: (2,1)
Move 9: (3,3)
Move 10: (1,4)
Move 11: (2,2)
Move 12: (4,3)
Move 13: (2,4)
Move 14: (1,2)
Move 15: (3,1)

Final board:
+---------------+
|  1| 14|  5| 10|
+---------------+
|  8| 11|  2| 13|
+---------------+
| 15|  4|  9|  6|
+---------------+
|   |  7| 12|  3|
+---------------+
Press any key to continue . . .
```

Program Style

Your program should follow the following coding conventions:

- `const` variable identifiers should be used as much as possible, should be written all in upper case and should be declared before all other variables;
- variable identifiers should start with a lower case letter, be meaningful, and be declared at the top of a function;
- every `if` and `if-else` statement should have a block of code (i.e. collections of lines surrounded by `{` and `}`) for both the `if` part and the `else` part (if used);
- every loop (`do`, `while`, `for`) should have a block of code;
- the keyword `continue` should not be used;
  the keyword `break` should only be used as part of a `switch` statement;
- opening and closing braces of a block should be aligned;
- all code within a block should be aligned and indented 1 tab stop (or 4 spaces) from the braces marking this block;
- global variables should be used sparingly with parameter lists used to pass information in and out of functions
- local variables (excluding loop counters) should only be declared at the beginning of functions (either as parameters or otherwise)
- commenting:
  - There should be a block of header comment which includes at least
    - file name
    - student names
    - student identity numbers
    - a statement of the purpose of the program
    - date
    - the percentage of the work completed by the authors — 50:50 is expected and assumed but reasons should be given if it is more/less than this
  - Each variable declaration should be on a single line and should be commented
  - There should be a comment identifying groups of statements that do various parts of the task
  - Comments should describe the strategy of the code and should not simply translate the C into English

Style marks will be awarded proportionally, i.e. if you attempt only half the coding you can expect only half the style marks.

Marking scheme

Task/Topic	**Maximum mark**
Program operates as specified	
`square_state.c` correctly completed	2
`game_state.c` correctly completed	10
`t_node.c` correctly completed	5
`game_tree.c` correctly completed	21
`stack.c` correctly completed	5
`queue.c` correctly completed	5
Program Style	
Does not unnecessarily repeat tests or have other redundant/confusing code	6

Uses correctly the C naming conventions	6
Alignment of code and use of white space makes code readable	6
Always uses blocks in branch and loop constructs	6
Meaningful identifiers	6
Variables declared at the top of functions	6
Each variable declaration is commented	6
Comments within the code indicate the purpose of sections of code (but DO NOT just duplicate what the code says)	6

What and how to submit

What to submit

- You should submit the entire Visual Studio project folder — or folder of source files if Visual Studio was not used — compressed as a ZIP file. *Do not submit a RAR archive — it will not be marked.*

How to submit

- Log in to MyLO and navigate to the `Assignments` tool under the `Assessments` icon in the top tool bar.
- Select `Assignment 3` from the list of available drop-boxes.
- Click on `Add a File` and follow the instructions to attach your ZIPped project folder and then click `Add`. Then click `Submit`.

If you want to re-submit, simply do so.

Please note: only one submission is required for the pair. You cannot submit as an individual only as a registered member of a pair. ***If you can't see the Submission drop-box, it's because you're not registered in a pair.*** *Fix that problem first.*

Plagiarism and Cheating:

Practical assignments are used by the School of ICT for students to both reinforce and demonstrate their understanding of material which has been presented in class. They have a role both for assessment and for learning. It is a requirement that work you hand in for assessment is your own.

Working with others

One effective way to grasp principles and concepts is to discuss the issues with your peers and/or friends. You are encouraged to do this. We also encourage you to discuss aspects of practical assignments with others. However, once you have clarified the principles *of the question*, you must express them in writing or electronically entirely by yourself in your pair. In other words you and your partner must develop the algorithm to solve the problem and write the program which implements this algorithm yourselves. You can discuss the question with others, but not the solution. Assistance with the solution should be provided by staff during tutorials or consulting times.

Cheating

- Cheating occurs if you claim work as your own when it is substantially the work of someone else.

- Cheating is an offence under the Ordinance of Student Academic Integrity within the University. Furthermore, the ICT profession has ethical standards in which cheating has no place.
- Cheating involves two or more parties.
  - If you allow written work, program print-outs, or electronic versions of your code to be viewed, borrowed, or copied by another student then you are *an equal partner* in the act of cheating.
  - You should be careful to ensure that your work is not left in a situation where it may be used/stolen by others.
- Where there is a reasonable cause to believe that a case of cheating has occurred, this will be brought to the attention of the unit lecturer. If the lecturer considers that there is evidence of cheating, then no marks will be given to any of the students involved and the case will be referred to the Head of School for consideration of further action.

Julian Dermoudy, May 6th 2021.