16. The Programming Project

A summary of this session's
Programming Project

About the Project

- It contributes 80% to the assessment of the module
  - The two online class tests contribute the other 20%
- You can work in pairs or individually
  - You cannot work in groups larger than this, and you cannot work jointly with other pairs or individuals
  - If you work as a pair, you must work on all parts of the project together

What you should submit

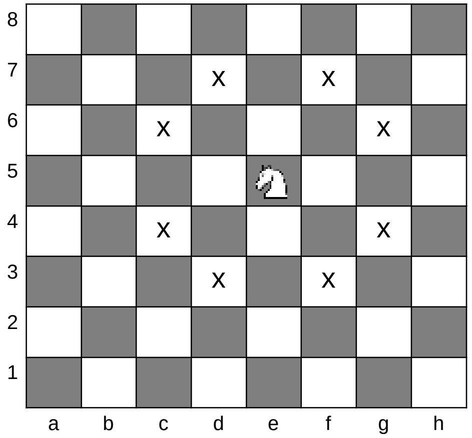
- Whether you work in a pair or individually, each of you should submit the project work on Moodle
- In addition to submitting the code for your program, well-commented to document it, each of you should write and submit an individual report on the work
  - The report is to be done individually, not in pairs
  - Your submission will not be accepted if you do not include an individual report of the work done

The Open Knight's Tour (or Knight's Path)

- A standard chess board has 64 squares on it, arranged in 8 rows called *ranks* (numbered 1 to 8) and 8 columns called *files* (labelled a to h)
  - The board is the same as that used in a game of draughts
  - Each square is identified by its file and its rank (e.g. a1, e5, h7)
- The Knight is one of the "minor pieces" in chess
  - In a standard pattern chess set the Knight is represented as a horse's head
- The Knight's move is L-shaped, and it is the only piece that can "jump" over other pieces
  - Squares on the board are alternately white and black (or light and dark) in colour and a Knight's move is such that if it is on a light square it moves to a dark square and *vice-versa*
- The open Knight's Tour problem asks the question, from some arbitrary starting square on an otherwise empty board, is there a sequence of moves that the Knight can make such that it visits every square on the board exactly once?

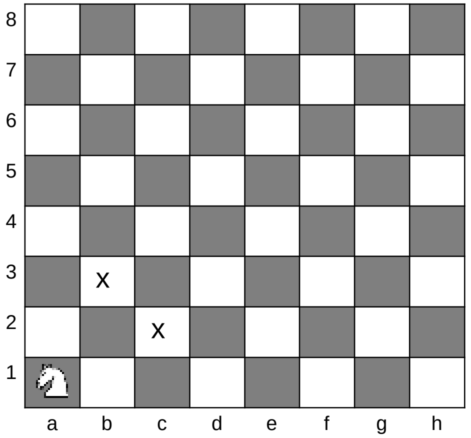
How the Knight moves

For a Knight at position (file, rank) it moves to a square that is one file and two ranks away from its current square, or two files and one rank away



How the Knight moves continued

A Knight in the centre of the board has eight squares it can move to, at the edge or in the corner of the board it has fewer available squares than this



The Open Knight’s Tour

- The mathematician, Euler, produced an analysis of the Knight’s tour problem in 1759, (though the problem is much older than this)
- It was not till the 1990s that it was proved that an open tour (one that does not require that the Knight is a move away from returning to its starting position at the end of the tour) is possible on all NxN boards where N >= 5

The Program

- The project is to write a console program that allows the user to find a solution to the open Knight’s tour by moving a Knight around the board
  - You are NOT being asked to write a program that solves the problem
  - You are NOT being asked to produce a graphical application
- To simplify running and testing the program the program should display a board with 3 ranks and 4 files
  - So the board has 12 squares that the Knight must visit
  - A tour on a 3x4 board is possible if the Knight starts in one of the outer files, you can use a1 as the starting square

The Program continued

- The program should be able to detect when the Knight has completed a tour
- The program should prevent any move that attempts to revisit a square that has already been visited
- The program need not detect when the user has reached a dead-end and the Knight has no available moves
  - That is, where all the squares that the Knight can reach have already been visited and there are still unvisited squares on the board
  - The user is expected to realise when this has happened and to quit at that point

A possible run of the program (user input is in bold)

```
Welcome to the open Knight’s tour problem! This program
allows you to move a Knight around a 3 by 4 chessboard. Your
task it to have it visit each of the 12 squares on the board
exactly once (i.e. without visiting any square twice).

An empty board would be displayed in the following format,
with “White” at the top:

      a   b   c   d
1:    -   -   -   -
2:    -   -   -   -
3:    -   -   -   -
```

```
As the knight moves, the sequence of moves is shown by
numbers in those squares it was or is occupying – the
highest number indicating where the knight is currently
located. For example, in the board below the knight is on
b1, having started at a1, moved to c2, then to a3, and then
to b1:

      a   b   c   d
1:    1   4   -   -
2:    -   -   2   -
3:    3   -   -   -

When referring to a square to move to, give the coordinates
in the standard chess algebraic format “a1” or “c3”. If you
run into a dead end and can make no further progress, type
“q” to quit.
```

```
The problem starts NOW. This is the initial position:

      a   b   c   d
1:    1   -   -   -
2:    -   -   -   -
3:    -   -   -   -

The Knight is on a1, where do you want to move the Knight to
(enter your move in the format <file><rank>, for example:
a1, or q to quit)? b3

      a   b   c   d
1:    1   -   -   -
2:    -   -   -   -
3:    -   2   -   -
```

```
The Knight is on b3, where do you want to move the Knight to
(enter your move in the format <file><rank>, for example:
a1, or q to quit)? d2

      a   b   c   d
1:    1   -   -   -
2:    -   -   -   3
3:    -   2   -   -

The Knight is on d2, where do you want to move the Knight to
(enter your move in the format <file><rank>, for example:
a1, or q to quit)? b1

      a   b   c   d
1:    1   4   -   -
2:    -   -   -   3
3:    -   2   -   -
```

```
The Knight is on b1, where do you want to move the Knight to
(enter your move in the format <file><rank>, for example:
a1, or q to quit)? a3

      a   b   c   d
1:    1   4   -   -
2:    -   -   -   3
3:    5   2   -   -

The Knight is on a3, where do you want to move the Knight to
(enter your move in the format <file><rank>, for example:
a1, or q to quit)? c2
```

```
      a   b   c   d
1:    1   4   -   -
2:    -   -   6   3
3:    5   2   -   -

The Knight is on c2, where do you want to move the Knight to
(enter your move in the format <file><rank>, for example:
a1, or q to quit)? q

OK. You have elected to quit – you have not found a tour of
the board. Good luck next time! Goodbye.
```

Comments on the program run

- At the end of the sequence in the above run, the Knight has reached a square where it cannot move to a position that it has not already visited
- So the user has not found a successful sequence and quits
- The project specification asks you to think about and make improvements to the functionality of the program and you will get extra marks for this
  - One possible improvement would be to allow the user to retract moves if they run into a dead-end
  - You can pass the project, however, with just the basic functionality asked for

The Marking Scheme

Point No.	Description	Possible Marks
1	The program follows the above specification in providing a description of the problem, the user instructions, and the initial state of the board (the Knight is initially on a1 in a 3x4 board).	10
2	The program defines a suitable set of classes and variables to represent the program state. This includes representing the board, where the Knight has visited and where the Knight and is currently located.	10

The Marking Scheme continued

Point No.	Description	Possible Marks
3	For each move, the program gets the user input and checks that, if the input is a move, that the move is a legal Knight's move and that it does not revisit a square that has already been visited. If the input is a request to quit the program displays a suitable message and exits.	10
4	For each valid move, the program updates the program state (i.e. moves the Knight) and displays the new program state.	10

The Marking Scheme continued

Point No.	Description	Possible Marks
5	After each move, the program checks if the Knight has completed the tour and if so it stops the sequence and displays a congratulatory message.	10
6	Additional functionality. Marks will be awarded for providing additional features relevant to the problem. Examples could include (examples on next few slides):	25

The Marking Scheme continued

Point No.	Description	Possible Marks
6 cont'd	a. Allow the user to choose the starting square b. Detect when a Knight's path will lead or has led to a dead end and alert the user to the situation c. Allow the user to retract their last move and try an alternative	25

The Marking Scheme continued

Point No.	Description	Possible Marks
6 cont'd	d. Allow the user to retract a series of moves and try an alternative to the earliest move in the retracted sequence, OR e. At the end of the game replay each move of the game's move sequence for the user to a point where the user would like to deviate from the previous sequence and try an alternative	

The Marking Scheme continued

Point No.	Description	Possible Marks
6 cont'd	f. Extend the program to deal with 5x5 or larger boards (in which case, the user could choose the starting square) g. Provide an option to save the Knight's path (the sequence of moves) to a text file for later analysis	

The Marking Scheme continued

Point No.	Description	Possible Marks
7	Program structure. The functionality of the program is distributed across a number of methods with appropriate parameters and return types, and/or across a number of classes each of which encapsulate some aspect of the problem or solution.	15

The Marking Scheme continued

Point No.	Description	Possible Marks
8	Report. The submission includes an individual report on the work done, how the program was developed and tested, and any areas where the program could be improved or further developed given more time and effort. **(Note: this report MUST be included or the submission will not be accepted – and a report must be produced and submitted by each of you individually even if you did the rest of the work as part of a pair).**	10

A few minor hints!

- Simplest approach is to represent board as a two-dimensional array of **`int`** with 3 rows and 4 columns
  - 0 means square unvisited, any other number is the move number that the square was visited on
- Input format for squares is standard algebraic chess notation (a1, c3, d2 etc)
  - See the handout on how to convert this format to array row and column index values
- Displaying the board
  - Remember you can line things up by using TextIO.put(x, y) – where x is displayed right justified in a field width of y characters, or by using TextIO.putf() (or System.out.printf())

Due date

- You should submit your project work as a zip file (just as with the practice project) by 4 pm on Friday 1st May (this is the end of week 12)
- You can work on the project in the lab sessions but you will need to work in your own time too
- Good luck!

Questions?