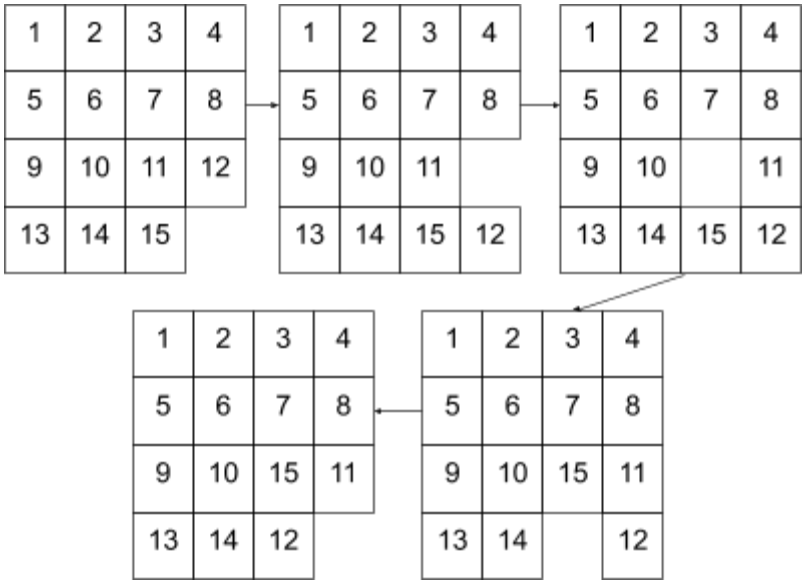
CSC8501 Coursework 2 – 2019
15-Puzzle (With More Extras)
Due 25th October 2019 at 10am

Set by Graham.Morgan@ncl.ac.uk



Specification (what you need to do): You will extend your computer program from coursework 1 to demonstrate strategies for simulating advanced features of the 15-Puzzle problem

Additional rules governing the simulation

- Include all the rules from coursework 1
- Partial continuous denoted with a constant numeric value indicates a shortened validity for orderings. For example, there are 11 2-partial continuous row orderings for the first configuration in the diagram from coursework 1 (repeated here for your convenience)

The Requirements

- Using a method of your choice, extend your solution from coursework 1 to identify all 3-partial continuous row and 3-partial continuous column (including reverse) in any given 15-Puzzle configuration
- Extend your solution to identify all 3-partial continuous row and 3-partial continuous column (including reverse) in all reachable 15-Puzzle configurations using valid “turns” from any given valid 15-Puzzle configuration
- Ensuring your user interface still allows manual and random 15-Puzzle generation, extend your interface to allow a user to request the finding of either 2, 3, or 4 -partial continuous row/column (including reverse) in any given 15-Puzzle configuration (as requested by a user - random or manual)

- Allow your user to request the finding of 2, 3, or 4 -partial continuous column/row (including reverse) 15-Puzzle configurations reachable using valid “turns” from any given valid 15-Puzzle configuration
- Enhance your Solution-File to add the values for partial continuous solutions (the numbers shown here may be indicative only and not a true representation of an accurate solution):

```
2
1     2     3     4
5     6     7     8
9     10    11    12
13    20    15
row = 2302
column = 2344
reverse row = 2341
reverse column = 2341
(total for row & column, including reverse, in this configuration)
2 = 22
3 = 14
4 = 8
(total for row and column, including reverse, for all valid turns)
2 = 235223
3 = 34563
4 = 456
```

- Check your solution for correctness with friends and colleagues in the class
- Update your solution to allow the user to specify the size of the 15-Puzzle problem (e.g., 25-Puzzle, 36-Puzzle, 49-Puzzle)

Deliverables (what we want to see submitted):

- C++ source code authored by the student
- Executable file containing solution
- A *15-file* containing 10 15-Puzzle configurations
- The associated Solution-File
- Demonstration (your chance to explain and show your solution):
    - On Friday 25th October from 10am onwards students will demonstrate their solutions

Learning Outcomes (what we expect you to demonstrate in a general way)

- To be able to design and create programs
- To be able to identify appropriate techniques for analysing the efficiency of programs
- To be able to realise inappropriate usage of programming languages
- To be able to manage memory
- To be able to create and use data structures

- To be able to use condition statements, loops and functions
- To be able to utilise concurrency when appropriate
- To be able to create programs that handle run-time errors
- To be able to use appropriate techniques for debugging and analysing existing algorithms
- To be able to design programs using a well known methodology

Marks Available (25):

Implementation gains up to 25 marks (correct working implementation guarantees 25 marks)

- **10 Marks for achieving correct output - Total =**
    - **1** Mark for sending an updated solution (partial) to screen or file (irrelevant of correctness); **1** Mark for sending multiple updated solutions (partial) to screen or file (irrelevant of correctness); **1** Mark for sending one valid updated solution (partial) to screen or file (correct in terms of 2 partial ordering); **1** Mark for sending multiple valid updated solutions (partial) to screen or file (correct in terms of 2 partial ordering); **1** Mark for sending one valid updated solution (partial) to screen or file (correct in terms of 3 partial ordering); **1** Mark for sending multiple valid updated solution (partial) to screen or file (correct in terms of 3 partial ordering); **1** Mark for sending one valid solution of a Puzzle where the user determines its size; **1** Mark for sending multiple valid solutions of Puzzles where the user determines their size; **1** Mark for sending multiple valid updated solution (partial) to screen or file (correct in terms of and partial ordering less than the size of a row/column in arbitrary sized Puzzles); **1** Mark for complete and whole solution!
- **5 Marks for appropriate input and output validation - Total =**
    - **1** Mark for filename validation; **1** Mark for file validation (is file present?) ; **1** Mark for reading files from other colleagues; **1** Mark for validating file output; **1** Mark for creation of enhanced Solution-File
- **3 Marks for user interface design - Total =**
    - **1** Mark for handling incorrect user input; **1** Mark for validating user input; **1** Mark for allowing choice of which element of the program to run (from interface)
- **2 Marks for parallel execution - Total =**
    - **1** Mark for threads; **1** Mark for threaded solution quicker than non-threaded solution;
- **5 Marks for advanced features - Total =**
    - Student can pick up **1** mark for each advanced feature and may include: Classes, Inheritance, Polymorphism, Lambda Functions, Compile time Optimizations; Pointers-to-Pointers; Templates; Assembler/Machine Code Augmentation