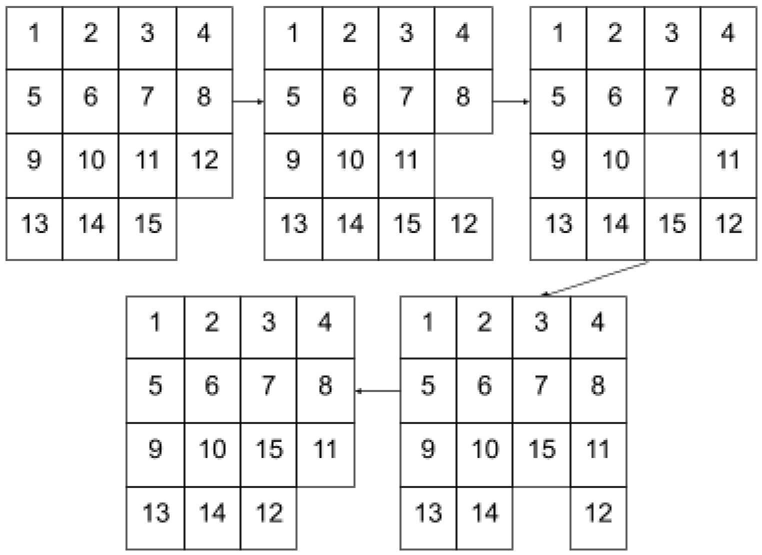
CSC8501 Coursework 1 – 2019
15-Puzzle (With Extras)
Due 25th October 2019 at 10am

Set by Graham.Morgan@ncl.ac.uk



Specification (what you need to do): You will build a computer program in C++ to demonstrate strategies for simulating a variation of the 15-Puzzle problem

The rules governing the simulation

- There is space for 16 blocks to be placed together in a square
- Only 15 blocks are present and are labelled with numbers (e.g., 1 - 15 in diagram)
- A "move" is represented as placing a block adjacent to the space into the space
- A "turn" is a number of "moves" with the space ending up in the bottom right corner
  - A "turn" is shown in the diagram
- A starting position ensures the space is always located in the bottom right hand corner
- After a "turn" there will be a number of blocks that are "continuous"
- Continuous row = numbers incremented by 1 from left to right for each column that has a block in it
  - All the rows are continuous in the first configuration of blocks in the diagram
- Continuous column = numbers incremented by 1 from top to bottom for each column that has a block in it
  - No columns are continuous in any of the configurations in the diagram
- Reverse continuous row = numbers incremented by 1 from right to left for each row that has a block in it
  - There exists no reverse continuous rows in any of the configurations in the diagram

- Reverse continuous column = numbers incremented from bottom to top for each column that has a block in it
    - There exists no reverse continuous columns in any of the configurations in the diagram
- Blocks can be labelled with numbers 1 through to 20 with no one block sharing the same number with any other block in the same configuration

The requirements

- Create a program that will allow a user to manually type in a 15-puzzle configuration using numbers 1 through 20 (inclusive)
    - Make sure not to allow repeated numbers for the blocks
- Create a program that will create 15-Puzzle configurations using numbers 1 through 20 (inclusive) in a pseudo random way
    - The number of these configurations can be chosen by the user
- Produce a text file (the 15-File) that stores 15-Puzzle configurations that your program generates
    - Text file format should start with a single number (indicating the number of 15-Puzzle squares) followed by the squares themselves (see below). Block labels should be separated by a space and each row should be on a new line

```
2
1      2      3      4
5      6      7      8
9      10     11     12
13     20     15

1      3      2      4
5      6      7      8
9      10     11     12
13     20     15
```

- Allow your program to read in a file (the 15-File) and deduce how many continuous rows, continuous columns, reverse continuous rows and reverse continuous columns are possible for all turns from the given configuration and print this to screen

```
1
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
```

- Allow your program to output its findings to a text file (the Solution-File) shown using the format above
- Compare your results with your friends and colleagues in class to ensure your program is correct
  - Read in the 15-Puzzle files of your colleagues and check their answers against your own (Solution-Files) (the numbers shown here in this document may be indicative only and not a true representation of an accurate solution)

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

- Be capable of designing and creating programs
- Realise inappropriate/appropriate usage of programming languages
- Understand how to manage memory
- To be able to create and use data structures
- To be able to use condition statements, loops and functions

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
- On Friday 25th October from 10am onwards students will demonstrate solutions

Marks Available (25):

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

- **10 Marks for achieving output - Total =**
  - **1** Mark for sending one 15-Puzzle to screen or file; **1** Mark for sending one 15-Puzzle in correct format to screen or file; **1** Mark for sending one valid 15-Puzzle to screen or file; **1** Mark for sending multiple 15-Puzzle to screen or file; **1** Mark for sending multiple valid 15-Puzzle to screen or file; **1** Mark for sending a Solution-File to screen or file; **1** Mark for sending a Solution-File to screen or file in correct format; **1** Mark for sending a valid Solution-File entry to screen or file; **1** Mark for sending multiple Solution-File entries to screen or file; **1** Mark for sending multiple valid Solution-File entries to screen or file
- **5 Marks for appropriate file input and output - Total =**

    - **1** Mark for opening a file; **1** Mark for closing a file; **1** Mark for sending valid output to file; **1** Mark for reading valid input from file; **1** Mark for creation of 15-Puzzle file and Solution-File
- **3 Marks for user interface design - Total =**
    - **1** Mark for prompting for user input; **1** Mark for allowing user input; **1** Mark for allowing program to execute again (from interface)
- **2 Marks for general solution - Total =**
    - **1** Mark for identifying continuous row and column numbers; **1** Mark for identifying reverse continuous row and column numbers
- **5 Marks for adhering to the rules of programming - Total =**
    - **1** Mark for use of functions; **1** Mark for use of appropriate variable identifiers; **1** Mark for understanding all code presented; **1** Mark for clean code (no commented out solutions and no `couts` apart from user interface and outputs of values); **1** Mark for optimisation (quality of solution)