Practical 7 (due: 2023-04-21 @ 09h00)

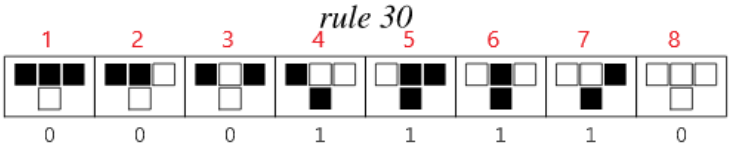
The purpose of this assignment is for you to get more proficient in working with dynamic one-dimensional arrays.

Please note: Submissions will be checked for originality. If you use someone else's code or code is taken from the Internet, then your prac will come under scrutiny for a potential copy, which may result in zero marks being awarded along with potential further disciplinary action.

Required knowledge for the prac.

Please make sure to review the tutorials regarding pointers and dynamic one-dimensional arrays as well as command-line arguments.

Cellular automata are used in many fields of computer science, physics, theoretical biology etc. One of the well-known cellular automata is known as the Rule 30 elementary cellular automaton. This version of the cellular automaton uses an array of elements that are considered to be either alive or dead. The array evolves into a new version of the array by applying the following rules for each element in the array:



We will assume edge cases wraps around. For example, if n==0, then we get the value of n-1 from the rightmost value.

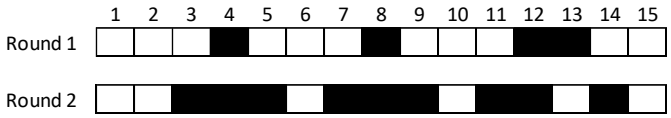


Figure 1: Example evolution of cells from round 1 to round 2. Black cells are alive. White cells are dead.

Figure 1 shows how the cells evolved between two different rounds. It demonstrates that element 3 in round 2 is alive because it matched the 7th pattern in round 1. Element 4 is alive in round two because it matched the 6th pattern etc.

Program

For this program, you must use developer-defined libraries with functions defined and declared in the ***CellularSpace*** namespace. Activities and functions in the program should use a dynamic one-dimensional array.

Write a C++ program that will accomplish each of the following requirements:

1. The program should use a menu system.
2. The size of the one-dimensional array must be provided using a command-line argument.
3. The user must have an option to define the number of rounds of evolution the cellular automaton will undergo when run.
4. The user must have the option to have random values assigned to the initial array.
5. The user must also have the option to set the initial values of the array manually.
6. The user must have the option to run the cellular automaton for the number of rounds specified in requirement 3. The program must output an easy-to-understand representation of the automaton as it evolves from one round to the next.
7. The user must have the option to quit the program.

Design

The design should cover the function that sets each value of the new generation array, given the values of the old generation array using the correct rules. Make sure to save your design as a PDF document.

Please format your practical for submission according to the structure established in the previous practical (`Docs` directory containing design, `Source` directory containing `*.cpp`, `*.h` files). The submission archive must be named according to the convention established in the previous practicals.

Please note: Programs that do not compile OR fail to link will be capped at 40%

Mark sheet		
	Program design	10
	Coding style (naming of variables(3), indentation (3), comments (2), const-variables(2))	10
	Functional abstraction (task decomposition (6), repetitive code (4))	10
	Separate compilation (Library files (6) and programmer namespace (4))	10
	User interaction (Menu system, appropriate input and output streams and feedback)	10
	Initialising, allocating and deallocating memory for the one-dimensional array.	10
	Setting the one-dimensional array with random values.	10
	Setting the one-dimensional array with manual values.	10
	Successfully generating the automaton from one generation to the next.	20
		/100