Practical 6 (due 2022-04-01 @ 09:00)

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

Binary Battle

The Utopian department of correctness requires a game to train their employees in conformity. The game should simulate a one-dimensional array of 20 random ones (1) and zeros (0). Below the list of ones and zeros a pointer should be displayed. The objective of the game is to teach employees to change each element in the array to either ones or zeros. This arrangement is depicted in Figure 1

0	1	0	0	1	0	1	0	0	1	1	1	0	1	0	1	0	0	0	1
									^										

Figure 1: Array of ones and zeros with pointer.

In the game you will need to move the pointer left or right to manipulate the ones and zeros. The value of an item changes its value from zero to one or one to zero when the pointer points to the item.

Initialisation:

- The size of the array is always 20 items.
- Each array item has a 50% chance of starting with either a zero or one value.
- The pointer is placed in a random location at the start of the game.

Movement:

- The player may move the pointer one step either left or right. The pointer may not move outside the array area.
- When the pointer moves, the value that the pointer points to changes its value. A value of one change to zero. A value of zero changes to one.
- The player may choose to quite the game at any time.

End game:

- The game ends in success if all the array elements have been changed to either ones or zeros.

For this program, you must use user-defined libraries with functions defined and declared in the ***BattleSpace*** namespace. Activities and functions in the program should use a fixed size one-dimensional array.

Design

The design should cover the function that initialises the initial state of the game. Make use of the design template provided in P01. 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 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
	Error handling.	10
	Initialising the start of the game.	10
	Handling the movement of the pointer	20
	Determining if the game was won	10
		/100