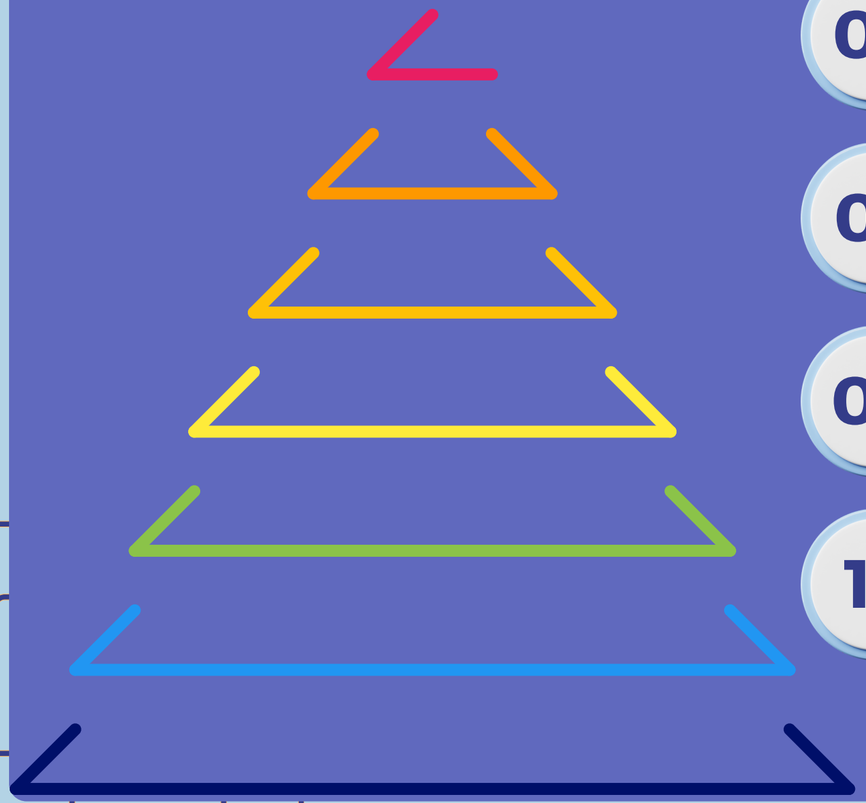
APPLIED PHYSICS 157

# ACTIVITY 10
# CELLULAR AUTOMATA

By : Manalang, Johnenn R.

# TABLE OF CONTENTS





# OBJECTIVES

## 01.

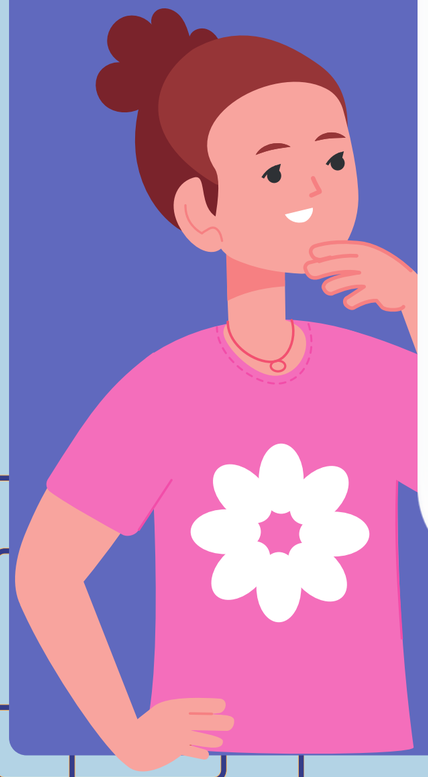
Define Cellular Automaton.

## 02.

In the context of CA, discuss determinism, randomness, and universality.

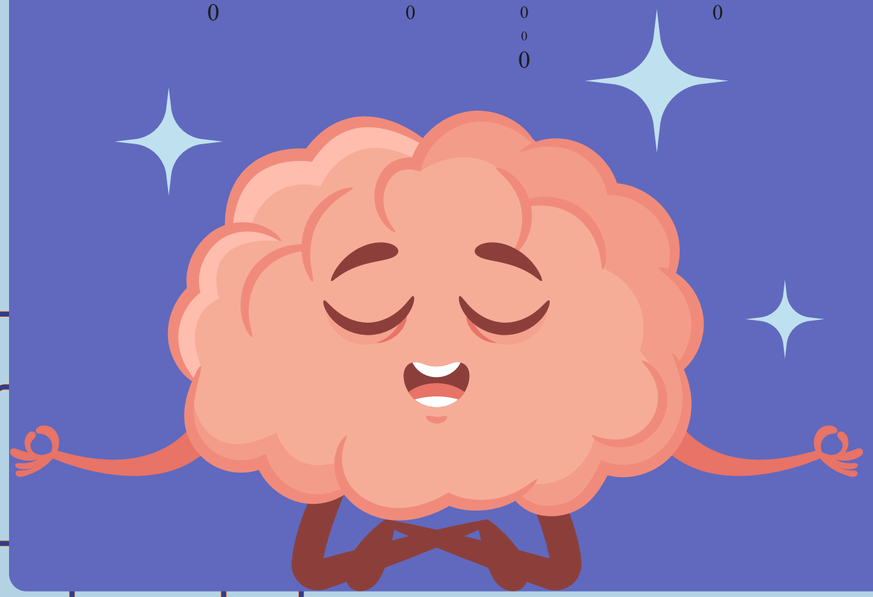
## 03.

Demonstrate Wolfram's 1D CA models.



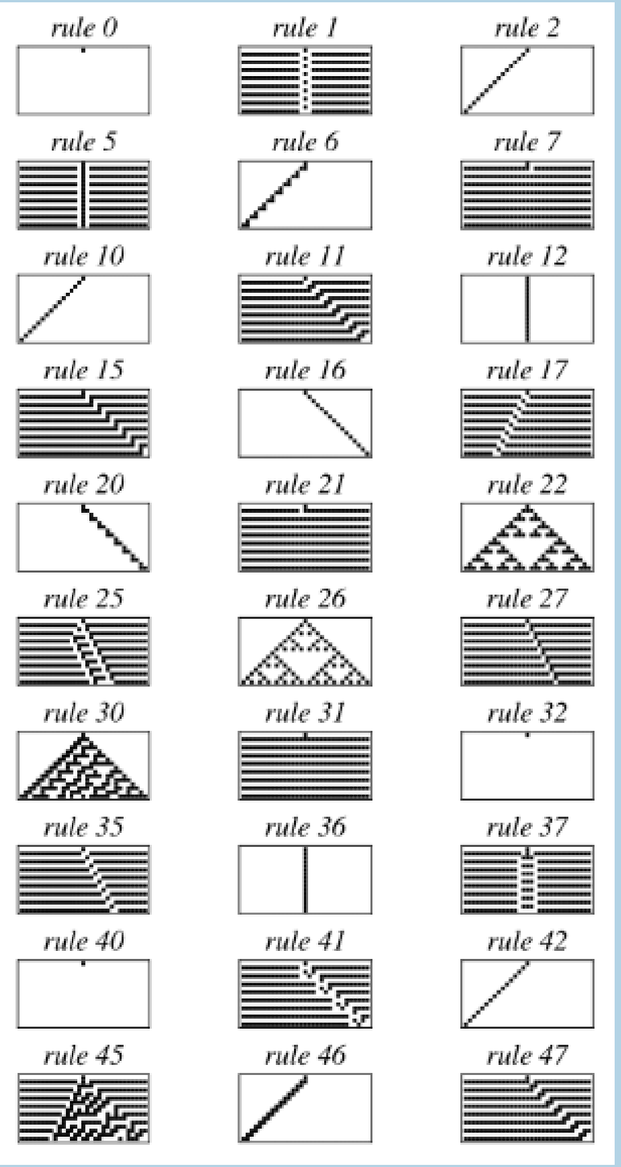
# OVERVIEW

Cellular automata (CA) are computational models consisting of a grid of cells whose states evolve based on predefined rules and neighboring cell states. They find applications in physics, biology, computer science, and social sciences. CA can model physical phenomena, study biological systems, solve optimization problems, simulate social dynamics, and more. Their flexibility and intuitive nature enable researchers to understand complex behavior and emergent properties in various fields.



# DETERMINISM, RANDOMNESS, UNIVERSALITY

In cellular automata (CA), determinism refers to the property where the evolution of a CA is entirely determined by the initial configuration and the defined rules. It allows for predictability and reproducibility. Randomness can be introduced through probabilistic rules or random initial conditions, adding unpredictability and the potential for emergent patterns. Universality is the property of some CA to compute any computable function, making them powerful computational devices capable of simulating any other computational system. These properties of determinism, randomness, and universality make cellular automata valuable tools for studying complex systems and computational phenomena.



# ELEMENTARY CELLULAR AUTOMATON

■ - 0 (OFF)

□ - 1 (ON)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| PREVIOUS | □ □ □ | □ □ ■ | □ ■ □ | □ ■ ■ | ■ □ □ | ■ □ ■ | ■ ■ □ | ■ ■ ■ |
| NEXT | ■ | ■ | □ | □ | ■ | ■ | □ | ■ |

In this activity, our goal is to showcase the Elementary Cellular Automaton, which represents the simplest form of 1D CA. These automata are characterized by having only two possible values (0 or 1) for each cell and their rules are solely dependent on the neighboring cells. The provided table illustrates the state that a particular cell will adopt in the next generation, based on the values of the cell to its left, its own value, and the cell to its right. Considering that there are 8 potential binary states for the three neighboring cells, there exist a total of 256 elementary cellular automata, each of which can be identified by an 8-bit binary number.

# CA1D FUNCTION

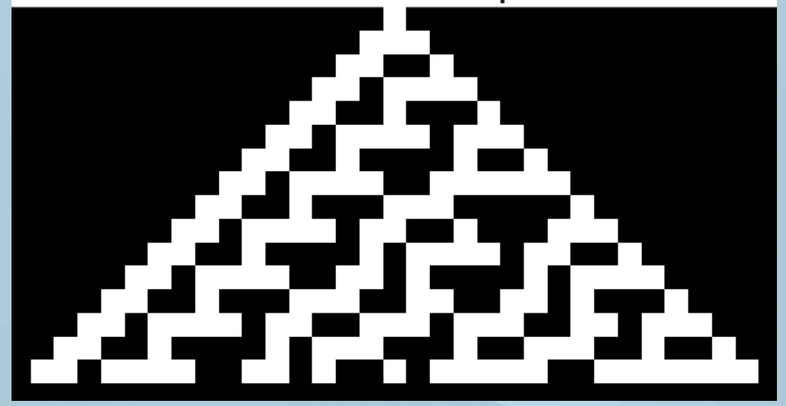
```
function a = CA1D(rule, timestep)
    % Convert decimal rule to 8-bit binary
    %then convert to numeric array
    ruleset = dec2bin(rule, 8) - '0';

    % Set the grid
    grid = zeros(timestep+1, timestep*2+1);

    %setting the "on" cell in the middle colum of the top row
    grid(1, timestep+1) = 1;

    %possible configurations of a 3-cell neighborhood
    %0 - on and 1 - off
    Prev = [1 1 1; 1 1 0; 1 0 1; 1 0 0; 0 1 1; 0 1 0; 0 0 1; 0 0 0];

    % Loop over timesteps/generations
    for i = 1:timestep
        a = imagesc(grid); colormap(gray);   % Display grid
        title(['CA Rule ' num2str(rule) ' after timesteps = ' num2str(timestep)]);
        axis image;
        axis off;
        drawnow;
        nextGrid = grid;

        % Loop over assigning value to next generation
        for j = 2:(timestep*2)
            neighborhood = grid(i, j-1:j+1);
            for k = 1:8
                if isequal(neighborhood, Prev(k, :))
                    nextGrid(i+1, j) = ruleset(k);
                    break;
                end
            end
        end
        grid = nextGrid;
    end
end
```

This function simulates a 1D CA with a specified rule and number of timesteps. The rule is converted from decimal to 8-bit binary representation, which is then used as a set of rules for the CA. The CA is represented as a grid, and the initial state is set with an "on" cell in the middle column of the top row. The function then iterates over the timesteps, displaying the grid at each step. It determines the next generation of cells based on the predefined rule and the current neighborhood configuration, updating the grid accordingly. The process continues until the specified number of timesteps is reached. Now, let's try to simulate some of the rules.

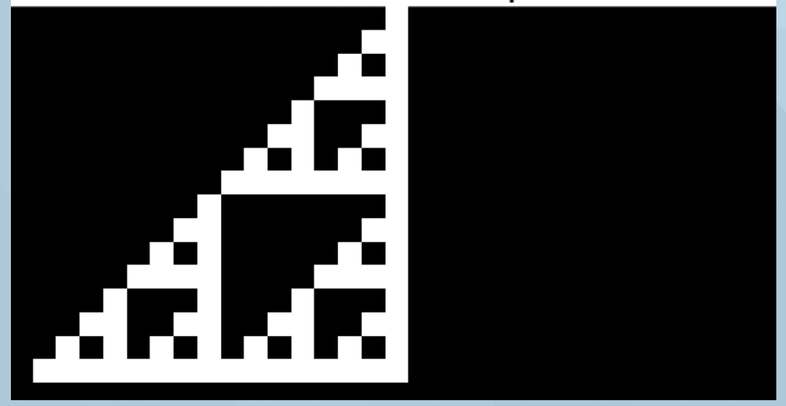
# SIMULATION OF DIFFERENT RULES

- 0 (OFF)
- 1 (ON)

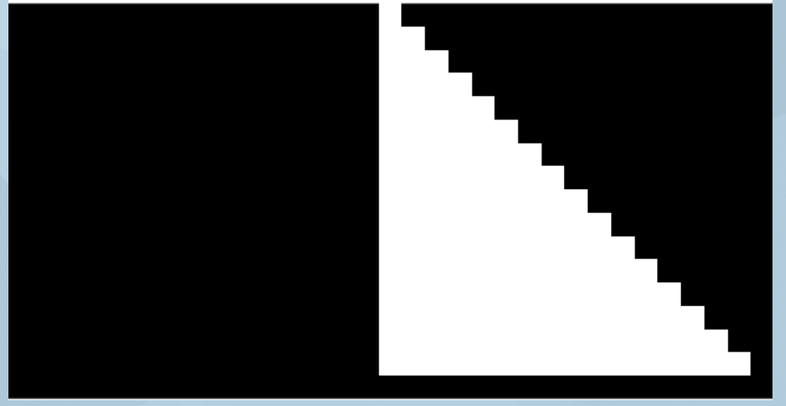
CA Rule 30 after timesteps = 16



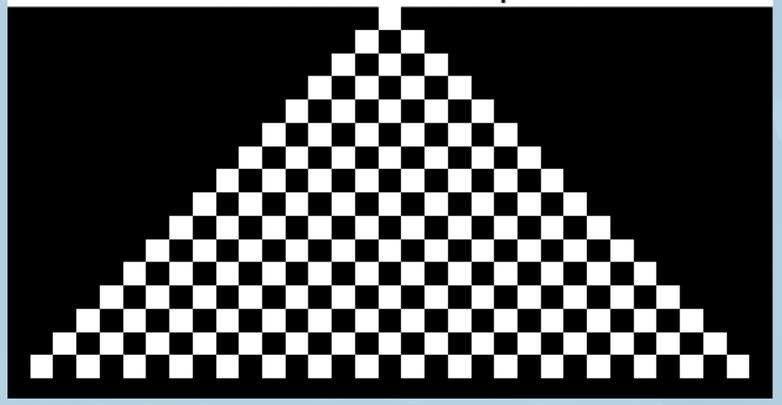
CA Rule 102 after timesteps = 16



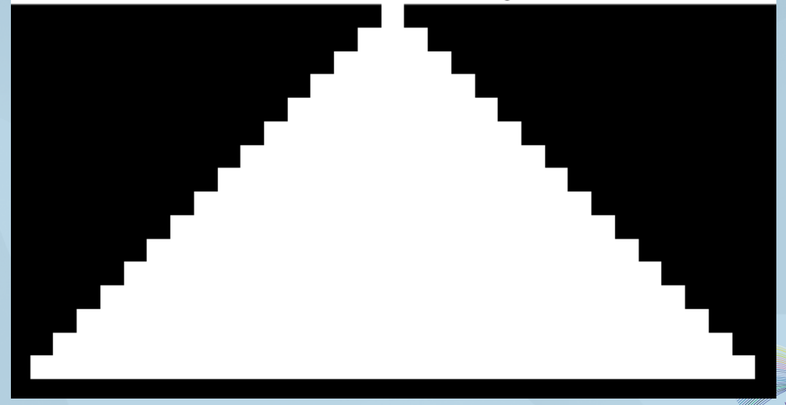
CA Rule 220 after timesteps = 16



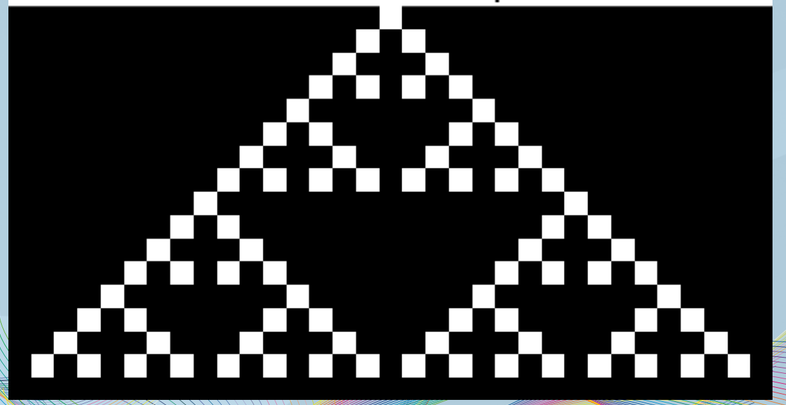
CA Rule 250 after timesteps = 16



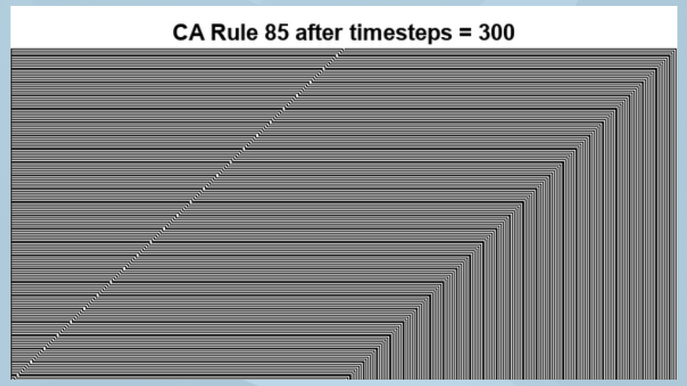
CA Rule 222 after timesteps = 16

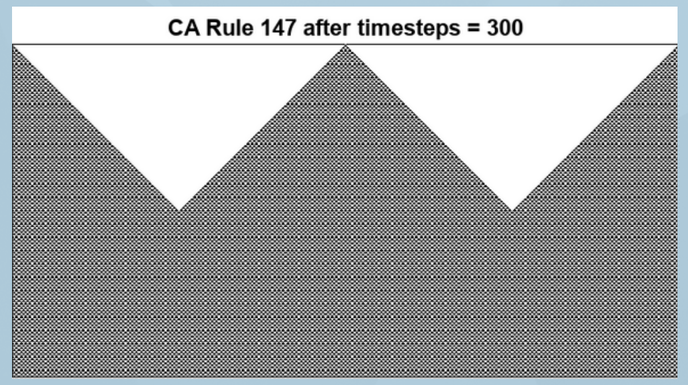


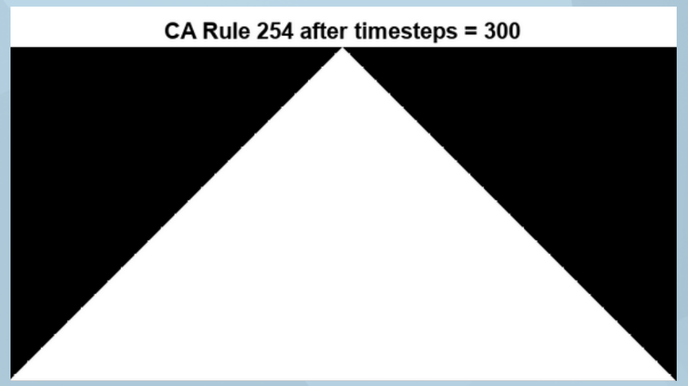
CA Rule 90 after timesteps = 16



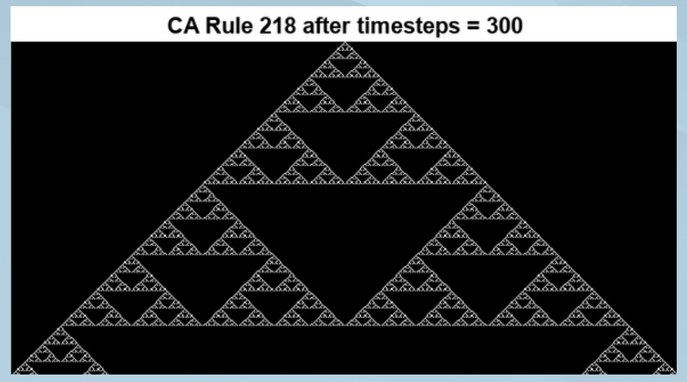
# SIMULATION WITH HIGHER TIMESTEPS

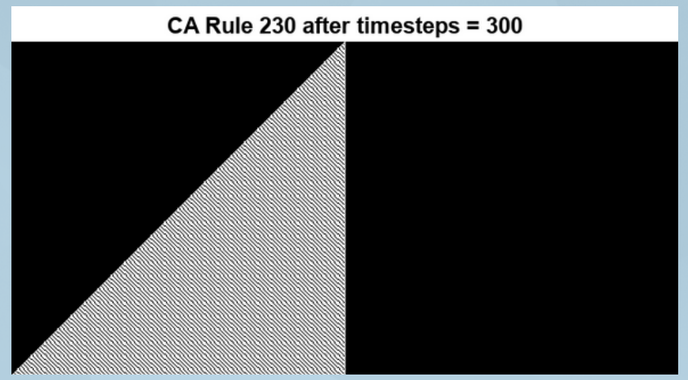


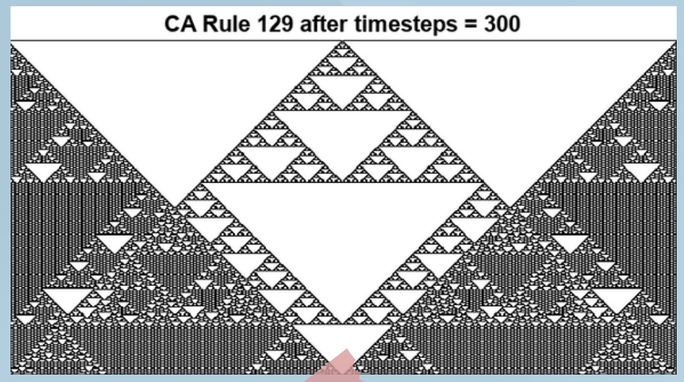




- 0 (OFF)
- 1 (ON)



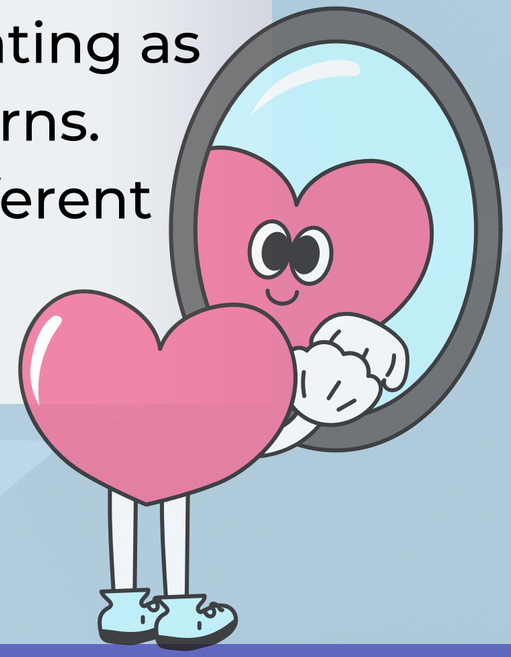




CA1D function successfully simulated the Elementary CA!

# REFLECTION

Creating Elementary Cellular Automata is not that difficult once you understand the concept behind it and how it works. It's a unique feeling of satisfaction when successfully simulating patterns similar to Wolfram's 1D CA. The algorithm is fascinating as it uses rules translated into binary to generate various patterns. Despite its simplicity, the applications of CA span across different fields, which is truly amazing.



# REFERENCES


1. https://uvle.upd.edu.ph/pluginfile.php/892120/mod_resource/content/1/thinkcomplexity2.pdf
2. https://mathworld.wolfram.com/ElementaryCellularAutomaton.html

# REPORT GRADE

| Criteria | Score |
|---|---|
| Technical Correctness | 35 |
| Quality of Presentation | 35 |
| Self Reflection | 30 |
| Initiative | 10 |
| TOTAL | 100 |

10

EVALUATION

Overall, I know that I have accomplished all the tasks included in this activity. I have successfully demonstrated how the Elementary Cellular Automata works and have shown several patterns in my report.