**Exercise 4.1**

I selected Set 1 with rules 73, 136, 184. The start situation is a black cell in the middle of the array.

These are the posible states

111 110 101 100 011 010 001 000

Depending on the rule used we define the outcome

**Rule 73: 1001001**

Interfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente256 257

Rule 136

**Rule 136: 10001000**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente 256 257

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Rule 184

Rule 136 don’t give any result except for the initial point, because all initial states, 010, 001, 100 and 000 gives an output of 0.

**Rule 184: 10111000**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente 256 257

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

In the first and third cases we can observe patterns. In the first case we can see how a complex pattern is created, where the first patterns begin to repeat in the following layers, doubling the patterns.

In the third case we obtain a simpler pattern where lines are created, separated in time. Those lines are generated because the state 100 gives a next state equal to 1, so the point on the right is going to be black. Moreover, when we get to the end of the array, another line appears because the point on the left is again 100.

I would say that rule 73 belongs to class 3 because it creates a complex pattern where is difficult to see periodicity, rule 136 belongs to class 1 because it doesn’t evolve, is stable and rule 184 belongs to class 2 because it can be seeing as an oscillatory pattern.

Finally, the length of the array doesn’t change the time of the repetitions of the patterns, but the change on the length of the array is only of 1 cell, so I will change the length a little bit more (150) in order to see if it is still the same.

In the first case is hard to know if there are any changes, but it seems like the same.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

In the third case, we can see how the pattern is repeated before, that is because the new line appears when the first line reached to the end of the array, so if the array is shorter, the generation of a new line will be faster.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Exercise 4.2**

In this exercise I selected “game of life” with two-dimensional array, and eight neighbors.

I have based my code on the one on this page <https://natureofcode.com/book/chapter-7-cellular-automata/>

1. **Death.** If a cell is alive (state = 1) it will die (state becomes 0) under the following circumstances.

**-Overpopulation:** If the cell has four or more alive neighbors, it dies.

**-Loneliness:** If the cell has less than two alive neighbors, it dies.

1. **Birth.** If a cell is dead (state = 0) it will come to life (state becomes 1) if it has exactly three alive neighbors (no more, no less).
2. **Stasis.** In all other cases, the cell state does not change. To be thorough, let’s describe those scenarios.

**- Staying Alive:** If a cell is alive and has exactly two or three live neighbors, it stays alive.

**- Staying Dead:** If a cell is dead and has anything other than three live neighbors, it stays dead.

Here we can see some captures of the evolution of the representation. Moreover, I attach the video to see it better, there we can see how the system evolve using those simple rules in a complex way. In these images the system reached an oscillatory state between two forms.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

.