**TEAM R**

Anas Saad (2510059)

Ahmed Youssef (2507690)

Ramin Hashemi (2508573)

**Terminal command:**

g++ main.cpp

**Report:**

I started by creating a main generation and a new generation which acts as a temporary holder for the next generations values. Then I created the function that displays one generation at a time, and the function that applies a rule only to a generation, the first problem I had was with how to represent the rule and how to apply it to the generation. I tried to do it efficiently 5 different ways but they all failed for some reason, so I decided to make the rule an array of 1s and 0s, ex: rule = {0, 0, 0, 1, 1, 1, 1, 0} I did it all with if and else if statement, with each containing a possible combination of cells and the executed code fills in the current index in the generation with the current index of the rule. So the only thing that needs to change is the rule array to contain the binary representation of whatever rule the users enters.

The second problem I had was with the function that takes in input for the rule in decimal form and converts it to its binary representation. The binary output was missing the leading zeros at the beginning, so I fixed this by adding zeros to the start until the loop reaches a 1.

The third problem I faced was with the user changing the length variable. The main and new generation arrays declarations are based on the constant integer variable LENGTH, but in order for the user to change the length, we have to remove the const keyword, but if we do, then there’ll be problems with the array’s declarations. There are multiple solutions that I had in mind, like declaring these arrays dynamically with pointers then deleting them after, but this would have required massive changes in the way the code is written. So after a lot of thinking and trial and error, I decided to create 2 new variables, LENGTH2 and lengthOffset. The LENGTH2 variable is based on the original LENGTH variable minus the lengthOffset. Then in the changeLength() function, it takes in a new length from the user then calculates the difference and the offset then sets LENGTH2 to the new length.

The fourth problem I encountered was that the first output when running the program was correct but then when running the same cell initiation mode with the same rule, the output comes out differently. After a lot of debugging, I figured out that it’s because the main generation is not resetting after the display, so I fixed this by resetting each of the memory locations of the main generation to 0, effectively resetting it back to it’s original state and making it ready for another output.