Report

Group members:

Viktoriia Yevsiukova - 2528502

Sujal Thakur - 2529583

Daniyal Abdullah Alvi – 2527355

**Introduction**

This report outlines the development of two distinct programs: a death game and a one-dimensional cellular automaton. The developers developed these programs for learning, illustrating, and modelling different elements of computer coding, such as rulemaking, interactive user, and simulating games. This report will discuss the design, operation and challenges encountered in the implementation of such programs.

**1D Cellular Automaton**

A one-dimensional Cellular automaton is a program for simulating the evolution of a one-dimensional cellular automaton following user-specified rules. Cells are arranged into generations where each generation is predetermined through an application of the rule in the preceding generation. The program was developed with several key aspects in mind:

**Storage for Generations:**

The efficient storage of generations comprised as one of the basic issues when developing the program. To simulate the evolution of cellular automata, two generations were utilized: the parent and child generations. A user defined rule on binary representations of parent generation served the purpose of generating a child generation. Having the different generations comprising our parents and children would therefore allow us keep true records on how our cellular systems might change in consecutive cycles.

**Rule Application:**

User defined rules formed the heart of the programs working. These rules defined each cell in the child generation depending on the content of parent generation. These rules could be defined either in an 8-bit binary string, or by simply indicating their respective rule numbers. Its versatility was also a factor as it enabled users to test out various rule sets and see how these affected the automaton’s operations.

**Generation Output:**

To provide users with a clear and visual representation of each generation, the program generated outputs on both the console and in a text file named "Automaton.txt". This approach allowed users to interactively observe the evolution of the automaton in real-time on the console while also providing a record of all generations in the text file. In the output, 'V' denoted living cells, while '.' represented the deceased cells, making it easy for users to interpret the results.

**Challenges Encountered**

Some of these problems were encountered during the design of one-dimensional Cellular Automata. These included:

*User Input Validation:*

* The user input for rules and generation length should be verified as valid and within the specified limits.

*Rule Application:*

* Efficiently applying the logic of user-defined rules using boundary conditions.

**Death Game**

The Death Game program simulates a cellular automaton where cells can be either alive ('V') or dead ('.'). The user interacts with the program by placing cells on a grid. The simulation then evolves these cells according to specific rules until the user decides to exit. Key features of the Death Game program include:

* **Cell Placement:** Users can specify the location of cells by entering row and column coordinates. These cells are initially placed on a grid.
* **Cell Evolution:** The program simulates the life and death of cells based on specific rules, which is a classic concept in cellular automata theory.
* **User Interaction:** Users can decide to continue the simulation or exit the program.

**Conclusion**

In conclusion, the tight teamwork and collaborative effort of our team resulted in a successful project. During the partnership, each individual received important expertise in specialised areas and learnt from others. The project not only displayed cellular automata's potential, but also our collaboration and problem-solving talents. We are excited to develop and refine this effort in order to investigate increasingly complicated automata and increase our knowledge of computational systems.

Word count: 580