Topic - Computing Systems Architecture

Series 13, 14, 15

November 2023

content

1 Administrative details

1.1 Deadlines

You can either send him the solutionsat the latest on January 8, 2024, at 23:59.

1.2 Reminder of the score on the topic

The theme is worth 40% of the grade in this laboratory (according to Course 0x00), and it is necessary to obtain a grade of 5 for passing.

1.3 Transmission

You will send the solutions in the following forms, depending on the series:

- series 13 (groups 131 and 132):https://forms.gle/koq26driB5ggJA1f8
- series 13 (groups 133 and 134):https://forms.gle/ymFukZ4LRikBnhaw8
- series 14:https://forms.gle/NxV4cDubT5eksWPR9
- series 15:https://forms.gle/koq26driB5ggJA1f8
- optional mathematics-informatics:https://forms.gle/cnLay24h5ULxXPJPA
- arrears:https://forms.gle/HXXz5PR1GLCp5dMi9

The links to the forms will be completed in the next period. Please come back to this document in the following days.

1.4 What will be transmitted

They will be uploaded in the formthree sources (one for each requirement) with the namesgroup - name first name 0.s, group first name first name 1.s,respectivelygroup name surname 2.s.If you have more than one name, you will upload named sources, for example,172 GeorgescuXulescu IonVasile 0.s.It isimportantto load sources with the correct name, because the testing will be automatic.

1.5 How the evaluation will be done

There are two steps to obtain note holding:

- all sources are checked to make sure there are no cases of plagiarism. If plagiarism is detected, a report is automatically made to *The Ethics Commission of the University of Bucharest*;
- sources that have passed the anti-plagiarism check will be tested automatically.

Important!Students who have different configurations than the ones we work on in the laboratory, must specify this in the form in which they submit the assignment, in order to be able to perform the evaluation and not to receive a default 0.

1.6 Other observations

1. We do not forbid you to discuss ideas among yourselves, but be careful, because there is an important difference between giving an idea and giving the code directly.
2. Do not use automatic converters from C/C++/other languages to x86, we have used them and we recognize without difficulty a code that is not written by you.

2 The formulation of the theme

Conway's Game of Lifeis a*zero-player game*two-dimensional, invented by the mathematician John Horton Conway in 1970. The purpose of this game is to observe the evolution of a system of cells, starting from an initial configuration, introducing rules regarding death, respectively the creation of a new cell in the system. This evolutionary system is*Turing-complete*.

The state of a system is described by the cumulative state of the component cells, and for these we have the following rules:

1. Underpopulation.Each cell (which is alive in the current generation) with less than two neighbors alive, dies in the next generation.
2. Continuity living cells.Each cell (which is alive in the current generation), with two or three neighbors alive, will also exist in the next generation.
3. Ultra popular.Each cell (which is alive in the current generation), which has more than three neighbors alive, dies in the next generation.
4. Creation.A dead cell that has exactly three living neighbors will be created in the next generation.
5. Continuity dead cells.Any other dead cell, which does not fit into the rule of creation, remains a dead cell.

The neighbors of a cell are considered to be the following 8, in a two-dimensional matrix:

A_{00}	A_{01}	A_{02}
A_{10}	the current cell	A_{12}
A_{20}	A_{21}	A_{22}

We define the state of a system at generation$n$as a matrix$S_n \in M_{m\times n}(\{0,1\})$ (m -the number of lines, respectivelyn -the number of columns), where element 0 represents a dead cell, respectively 1 represents a living cell (in the current generation).

We define ok-evolution ($k \geq 0$) of the system one iteration$S_0 \rightarrow S_1 \rightarrow \cdots \rightarrow S_k$, where each$S_{and+1}$is obtained from$S_{and}$, applying the five rules stated above.

Observation.For cells located on the first line, first column, last line, respectively last column, expansion to 8 neighbors is considered, by considering those thatnotthey are in the matrix as dead cells.

Example.Let the following be the initial configurationS_0:

0	1	1	0
1	0	0	0
0	0	1	1

First, we will consider the expansion of this matrix$S_0$of dimensions$3x4$in an extended matrix$\overline{S_0}$ of dimensions$5x6$,so:

0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	0	0	1	1	0
0	0	0	0	0	0

In what follows, we will work only inside the main matrix, but considering the expansion for the correct processing of the neighbors. We will go through each element, and we will see which evolutionary rule we can apply. For example, for the element at position (0.0)in the initial matrix, we will apply the dead cell continuity rule, because it is a dead cell and does not have exactly three living neighbors.

The next cell is alive, and has exactly two neighbors alive, so the rule of continuity of living cells applies.

For the cell at position (0.2)in theS_0, we notice that it has only one neighbor, so the underpopulation rule applies - the cell will die in the next generation.

Following the same reasoning for all cells, the configuration of the system in one iteration (in$\overline{S_1}$) will be:

0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0

Symmetric encryption scheme.We define an encryption key (starting from an initial configuration$S_0$and o k-evolution) as the operation$< S_0, k >$, which represents the paintingdimensionalof data (understood as a string of bits) obtained after concatenating the lines in the matrix from the obtained extended matrix,$\overline{S_k}$.

For example, starting from the previous configurationS_0, and applying only a 1-evolution, the extended matrix is obtained$\overline{S_1}$previously described, which will have the effect of applying the operation$< S_0, 1 >$obtaining the following one-dimensional table (understood as a string of bits):

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

consider$m$a clear message (a string of characters without spaces). Encryption$\{m\}_{<S_0,k>}$will mean XORing the message in the clear$m$with the result given by$< S_0, k >$. There are the following cases:

- if the message and the key have the same length, it is XORed element by element, until the result is obtained;
- if the message is shorter than the key, only the first part of the key is used, corresponding to the length of the message;
- if the message is longer than the key, the replication of the key is considered whenever it is necessary to encrypt the entire message.

We consider that$m$=password,and use as a key$< S_0, 1 >$, where$S_0$is the initial configuration described previously. We saw that the obtained result is the string of bits:

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

which we will consider without spaces:

000000001000000010000000000000

To perform the encryption, we must analyze the string to be encrypted, namelypassword.We will see what is the ASCII (binary) encoding of each character in this string:

p	01110000
A	01100001
R	01110010
a	01101111
it	01101100
A	01100001

Syrianpasswordwill thus be the binary string

011100000110000101110010011011110110110001100001

We notice, in this case, that the string to be encrypted is longer than the encryption key, so if we try an XOR now, we would have the following situation:

message = 011100000110000101110010011011110110110001100001
key = 00000000100000001000000000000

We will consider, in this case, that we will concatenate the key to the initial key again:

message = 011100000110000101110010011011110110110001100001
key = 0000000010000000100000000000000000000000100000001000000000000

And then we will keep only enough of the new key to encrypt the message:

message = 011100000110000101110010011011110110110001100001
key = 000000001000000010000000000000000000000001000000010

The encrypted message will be obtained by XORing element by element, knowing that0 XOR 0 = 1 XOR 1 = 0,respectively0 XOR 1 = 1 XOR 0 = 1.In this case,

message = 011100000110000101110010011011110110110001100001
key = 000000001000000010000000000000000000000001000000010 crypt
= 011100001110000111 110010011011110110111001100011

The displayed encrypted message will be in hexadecimal (to avoid problems displaying characters), and in this case we will have:

crypt = 0111 0000 1110 0001 1111 0010 0110 1111 0110 1110 0110 0011
= 7 0 E 1 F 2 6 F 6 E 6 3 = 0x70E1F26F6E63

For decryption, the same mechanism is applied, the decrypted message will be XORed with the calculated key, and we will finally havem XOR k XOR k = m. (k XOR k = 0,andm XOR 0 = m,from the associativity of XOR, respectively from the calculation rules). When decrypting, the message will not be displayed in hexadecimal, but in clear.

3 Requirements

In this theme you have three requirements - a requirement for 5p, one for 2.5p, respectively a requirement for another 2.5p.

Important! NOTgive manual input at each retest of the program! They are long inputs that will cost you time. Create a file, for exampleinput.txt,in which you write the desired input, and after you have an executable, for exampletask00,which you would normally run with ./task00, run the command ./task00 < input.txt.Thus, the content of the file will be redirected toSTDIN, exactly as when you entered the values manually. Use this information to test more inputs, creating files input0.txt, input1.txtetc., and testing them with ./task00 < input0.txt, ./task00 < input1.txtAnd so on

Important!All character strings (used for display) that you define in the section . datethey will have, in the end, the character \n!

Important!Requirements 0x01, respectively 0x02NOTthey are cascaded, so they can be solved independently.

3.1 Requirement 0x00 - 5p

They are read from the keyboard (STDIN)the number of linesm,the number of columnsno,the number of live cells pthe positions of the live cells in the matrix, respectively an integer*k*.Careful!In reading the input, the initial matrix is considered,unextended:the initial configuration is readS_0, andNOT$\overline{S_0}$! For example, for the matrix in the requirement statement, the input would be:

```
3                    // m - the number of lines // n -
4                    the number of columns // p - the
5                    number of live cells
0
1                    // first live cell is in (0,1)
0
2                    // the second live cell is in (0,2)
1
0                    // the third living cell is in (1,0)
2
2                    // the fourth live cell is in (2,2)
2
3                    // the fifth living cell is in (2,3) // the
5                    integer k
```

It is required, at this step, to display atSTDOUTof the system configuration after o*k*-evolution.Careful! The system status will be displayed$S_k$andNOTextended matrix$\overline{S_k}$!

The matrix will be displayed as usual, and in this case, the result is:

```
0 0 0 0
0 0 0 0
0 0 0 0
```

(all cells die after the second iteration).

Observation:The elements on the line will be displayed with a space between them, and at the end of each line, you will display a character \n.And after the last line you will display that character \n!

The following are guaranteed:

- 1≤m≤18
- 1≤n≤18
- *p≤n·m*
- *k*≤15

3.2 Requirement 0x01 - 2.5p

They are read from the keyboard (STDIN)the number of linesm,the number of columnsno,the number of live cells pthe positions of the living cells in the matrix, an integerk,an entireawhich can be 0 or 1 (0 for encryption, 1 for decryption), respectively a messagemwhich can be in the clear, for encryption (a string of the form0x...,for decryption). Encryption/decryption of the received message is requested, according to the key that must be calculated, according to the specifications in the theme formulation.

An input could be:

```
3                          // m - the number of lines // n -
4                          the number of columns // p - the
5                          number of live cells
0
1                          // first live cell is in (0,1)
0
2                          // the second live cell is in (0,2)
1
0                          // the third living cell is in (1,0)
2
2                          // the fourth live cell is in (2,2)
2
3                          // the fifth living cell is in (2,3) // the
1                          integer k
0                          // ENCRYPTION will be performed // the
password                   clear message to be encrypted
```

In this case, the result is0x70E1F26F6E63.

For input:

```
3                          // m - the number of lines // n -
4                          the number of columns // p - the
5                          number of live cells
0
1                          // first live cell is in (0,1)
0
2                          // the second live cell is in (0,2)
1
0                          // the third living cell is in (1,0)
2
2                          // the fourth live cell is in (2,2)
2
3                          // the fifth living cell is in (2,3) // the
1                          integer k
1                          // DECRYPTION will be performed //
0x70E1F26F6E63             the hex string to be decrypted
```

the result ispassword.

Observation.The same conditions are guaranteed as in the case of the first requirement, namely the correctness of the input data is guaranteed, the messages to be encrypted will be messages without spaces, made up of alpha-numeric characters (numbers, lowercase letters, uppercase letters), and the messages to be decrypted will be strings hexadecimals that will start with0xand will contain symbols from the crowd{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. The messages (considered in the clear) will have a maximum of 10 characters!

3.3 Requirement 0x02 - 2.5p

To restore, in a separate source file (named according to section 1.4.), the request 0x00, so that the input is read from a filein.txt,and the output should be written in a fileout.txt,using functions from the C language.

4 Concrete inputs

In this section, we will rewrite the inputs without the associated comments, to be clear in the form in which you will receive them.

4.1 Requirement 0x00

Input	Output
3	0 0 0 0
4	0 0 0 0
5	0 0 0 0
0	
1	
0	
2	
1	
0	
2	
2	
2	
3	
5	

Table 1: Input Request 0x00

4.2 Requirement 0x01

Input	Output
3	0x70E1F26F6E63
4	
5	
0	
1	
0	
2	
1	
0	
2	
2	
2	
3	
1	
0	
password	

Table 2: Input Requirement 0x01 - encryption

Input	Output
3 4 5 0 1 0 2 1 0 2 2 2 3 1 1 0x70E1F26F6E63	password

Table 3: Input Request 0x01 - decryption

4.3 Requirement 0x02

They are the same inputs as in Request 0x00, only that they will be read by the program from the filein.txtand then displayed inout.txt,using functions from the C language.