C PROGRAMM ABOUT CELLS

In a living cell, reactions of various kinds take place all the time. Each reaction involves different substances present in the cell and produces new substances (for example, proteins).

We can give a mathematical model of reaction like this: we fix a set

finite S of "substances"; a reaction is determined by three sets of substances:

- the set R of *reactants*
- the set I of *inhibitors*
- the set P of *products*

We describe the state, or configuration, of the cell at a certain instant by specifying a subset of S, $Q \subseteq S$. The elements of Q are the substances present in the cell at that instant.

A reaction $\alpha = (R, I, P)$ is *enabled* in Q if all its reactants are present in Q and no inhibitors

is present in Q.

An enabled reaction "happens" and has the effect of consuming the reagents and producing its own products, changing the state of the cell.

If in a configuration several reactions are enabled, we will assume that they all occur simultaneously. The effect of a set of reactions is the union of the effects of the individual reactions.

The next configuration will be formed from the union of their products.

You are asked to write a program that simulates the evolution of a reaction system starting from an initial configuration.

For simplicity, let's assume that each reaction has two reagents, an inhibitor, and two products, that the set of substances contains N elements, and that there are K reactions, with N and K set in #define directives.

The substances are identified by integers 0,1, ⋯, N - 1.

The program must read the initial configuration from standard input, in the form of list of integers on the first line.

Each subsequent line describes a reaction, such as in the example below:

```
a 2 4 3 0 2
```

The first character of the line is the reaction identifier; the following numbers match,

in order, to the two reagents, the inhibitor, and the two products.

The program must perform a simulation by calculating, starting from the initial configuration, which reactions are enabled and determining the next configuration.

The number of simulation steps to be performed must be passed to the program as an argument on the command line.

The program must print the configurations produced, one per line, calculate how many

times, during the simulation, each reaction takes place, and print the count values.

SOME CLARIFICATIONS

The purpose of the code is to represent a series of reactions that take place inside the cell.

Initially I define, with #define at the beginning of the code, the number of substances inside the cell N. N can take on any natural number greater than 0. In the same way, always using #define, I define K number of types of reaction that will occur at interior of the cell.

Once K and N have been defined, the program must read from the standard input the initial substances with which the user decides to start. Obviously, the number of substances chosen by the user will be < of the N initially defined. If N of the #define is equal to 5, it means that the elements available inside the cell will be 0,1,2,3,4. Then the user in the standard input can choose among these written above those with which to start in the initial configuration. Once the user has defined in the standard input the substances to start with (which correspond to numbers from 0 to N-1), in the lines below he will define the reactions he decides take place inside the cell. He defines them as follows: a 1 2 3 4 5. Where "a" is the name of the reaction, 1/2 the reactants, 3 the inhibitor and 4/5 the products. It can define up to a maximum of K reactions.

The initial state is therefore that defined in the standard input by the user. To understand which reactions take place from the first configuration to the next one, I have to see if in the initial configuration there are the reaction reagents and if the inhibitor is not present. In the second state (second configuration) I will therefore have as elements only the products of the reactions that were possible given the elements present in the initial configuration. The set of products alone forms the new state of the cell. We repeat the same procedure, but this time the elements available for the reactions are those contained in the second state, the one with the products of the first reactions.

it is important that the program prints all the configurations. Whenever new reactions occur and a new configuration is created with the new products, the program must print this new configuration.

Let's take an example to clarify everything. Suppose we have 3 defined reactions:

1. **a 1 3 4 2 8**
2. **b 3 2 6 4 5**
3. **c 5 6 7 8 3**

And suppose that the configuration with the initial elements chosen in the standard input by the user is 1 3 6 5. In this case the reaction a can occur because there are its two reagents and there is no inhibitor. Reaction b cannot occur because it does not have both reactants and there is the inhibitor and reaction c can occur because there are its two reactants and there is no inhibitor. In configuration 2 there will therefore be 3 8 2. With these elements the only possible reaction is reaction b which can now take place. The third configuration will therefore be 4 5. At this point there are no more possible reactions and therefore there are no further configurations. As you can see, if two reactions give the same element as a product, this appears only once in the configuration that is created. Obviously, all reactions from one state to another are supposed to occur simultaneously.

The number of simulation steps to be performed must be passed to the program as an argument on the command line. This means that if the program ends when either the number of configurations reached is equal to the number set by the user in the command line, or (as in the example proposed above), an equilibrium is reached in which reactions are no longer able to occur.

IMPORTANT REQUESTS

1. I kindly ask you to comment all the passages next to the corresponding line.
2. The code must be written in ANSI C.
3. It is not allowed to declare vectors whose size is specified by the name of a variable (it is possible to dynamically allocate space for a vector or for a matrix whose size is not known a priori). The program will compile with gcc and with the options -Wall -std = c99 and, if necessary, -lm.