SORTING ALGORITHM USING CHEMICAL COMPUTATION

Submitted by :

Pankaj Bhambhani (200901047)

Soumik Pal (200901024)

Santosh Kumar (200901046)

Submitted to:

Prof. Manish K. Gupta

PREFACE

This user manual includes all the reactions along with explanations that have been used in our program. Since our project was "SORTING ALGORITHM WITH CHEMICAL COMPUTATION ", we required numerous set of chemical computing constructs like for increment, decrement, comparing etc.

This manual will help you to understand the reactions involved in our project –please also refer to our submitted program for testing with valid input to get the output.

We have given the instructions to compile and run the program at the end of this document.

We would like to acknowledge the help and support of our Professor Manish Kumar Gupta, without whom this project would have been a mere fantasy.

- *Authors*

Copier :

The reactions for the copier construct are as follows. Firstly, in the presence of g, a reaction transfers the quantity of a to a':

$$\mathbf{g + a \rightarrow g + a` ;}$$

Secondly, after all molecules of a have been transferred to a', the system removes all the molecules of g:

$$g + a(abs) \rightarrow \varnothing ;$$

Here, again, we are using the concept of an absence indicator. Removing g ensures that a is copied exactly once.
After g has been removed, a reaction transfers the quantity of a' back to a and also creates this same quantity of b:

$$g(abs) + a` \rightarrow a + b;$$

Alternatively, we can use a slight modification of this reaction to double the quantity of a:

$$g(abs) + a` \rightarrow 2a ;$$

Increment Operation :

Given molecules of g, a reaction transfers molecules of x to molecules of x':

$$x + g \rightarrow x` + g \;;$$

The following reaction sets the quantity of g to zero.

$$g + x\,(abs) \rightarrow \emptyset \;;$$

Again, using the absence indicator mechanism, it proceeds only once all molecules of g have been removed:

$$g\,`(abs) + 2x\,` \rightarrow x + x` + \boldsymbol{x^{rx}} \;;$$

we do not directly use an absence indicator for g(abs), but instead, we use a secondary absence indicator g' (abs) .
This reaction also produces molecules of a supplementary type $\boldsymbol{x^{rx}}$. Note that this reaction is in the ‘‘fast’’ category. The new type $\boldsymbol{x^{rx}}$ is consumed by the reaction :

$$\boldsymbol{x^{rx}} \rightarrow \emptyset \;;$$

Finally, we include the following reaction to perform a decrement :

$$\boldsymbol{x^{rx}} + \boldsymbol{x}` + \boldsymbol{g}`(\boldsymbol{abs}) \rightarrow \emptyset \;; \quad [\text{ increment }]$$

Comparator :

Using our copier construct, we can create a construct that compares the quantities of two input types and produces an output type if one is greater than the other. For example, let us assume that we want to compare the quantities of two types a and b:

```
if (a > b)
 {
      t : = TRUE
else{
      t : = FALSE
 }
```

If the quantity of a is greater than the quantity of b, the system should produce molecules of an output type t; otherwise, it should not produce any molecules of t .

First, we create temporary copies, $\boldsymbol{a^c}$ and $\boldsymbol{b^c}$, of the types that we wish to compare, a and b, using the copier construct . We split the start signal so that the two copiers are not competing for it:

$$g \rightarrow \boldsymbol{g^1} + \boldsymbol{g^2} ;$$

Now we compare a and b via their respective copies $\boldsymbol{a^c}$ and $\boldsymbol{b^c}$.
To start, we first consume pairs of $\boldsymbol{a^c}$ and $\boldsymbol{b^c}$:

$$\boldsymbol{a^c} + \boldsymbol{b^c} \rightarrow \emptyset ;$$

We assume that this reaction fires to completion. The result is that there are only molecules of $\boldsymbol{a^c}$ left, or only molecules of $\boldsymbol{b^c}$ left, or no molecules of $\boldsymbol{a^c}$ nor $\boldsymbol{b^c}$ left. Molecules of the type that originally had a larger quantity have persisted. If the quantities were equal, then both types were annihilated. We use absence indicators $\boldsymbol{a^c}$(abs) and $\boldsymbol{b^c}$(abs) to determine which type was annihilated. If a was originally greater than b, there will now be a presence of $\boldsymbol{a^c}$ and an absence of $\boldsymbol{b^c}$. We produce molecules of type t if this condition is met. We preserve the quantities of $\boldsymbol{a^c}$ and $\boldsymbol{b^c}$(abs). We can also limit the quantity of t produced by introducing a fuel type :

fuel + $\boldsymbol{a^c}$ + $\boldsymbol{b^c}$ **(abs)** $\rightarrow$ $\boldsymbol{a^c}$ + $\boldsymbol{b^c}$ **(abs) + t ;**

Initialization (Take variable to zero) :

$$a \rightarrow \varnothing ;$$

This reaction is equivalent to assigning the value of zero to variable “a”

$$a = 0 ;$$

How to compile and run the program

To compile the program, use the following command on your terminal

```
javac SelectionSort.java
```

To run the program, use the following command on your terminal

```
javac SelectionSort.java
```

To vary the input parameters, see the `main()` function in the file – you can create array with different elements.