Preconceptions of novice learners about program execution

Sylvia da Rosa
Instituto de Computación
Facultad de Ingeniería - UDELAR
darosa@fing.edu.uy

Abstract
The ontological problem of the nature of programs is fundamentally based on the question "what is a computer program?" On the one hand, the algorithm is the *abstract* part of a program (the text) and on the other hand, the organization of data and the execution of instructions by a physical device (the machine), are the *concrete* part of the dual nature of programs. This question has important educational implications. In this paper we describe activities carried out to investigate novice students' preconceptions about how a program is executed by a computer.

1. Introduction
In this paper we describe the first stage of a research study of novice students' preconceptions about how a program is executed by a computer. The research is part of a project aiming to promote the exchanges between the field of Teacher Training and areas of scientific and technological research. The project seeks to enrich teachers in their professional role. For computer science this is a key issue considering the increasing importance of learning programming concepts at earlier levels. Especially, it is noted that in teacher training programmes the most important contribution should come from Computer Science departments in universities [2,3]. The activities were carried out with two groups of students. The first group was aged 13 to 15 with the activities taking place at *Liceo La Paz 2* (high school La Paz 2). The second group was aged 16 to 18 and the activities took place at *Instituto Tecnológico Informático (*Institute of Computer Technology*)*. Both locations are in Uruguay. The teachers of both groups took part in the facilitation of the activities within their classrooms. The content of the research activities was proposed by our research group in computer science education from the *Instituto de Computación* of the University (Computer Science Institute).

The aim of this research is to gather information about how student's preconceptions relate to their experience of simulating a computer running a program. To limit the scope of the investigation at this early stage, it is proposed to use a simple program to exchange the values of two variables (memory cells) using the assignment statement. In the activities the students were asked to:

- simulate the exchange using concrete material (glasses with liquids),
- write an algorithm for the exchange (in natural language),
- simulate the behaviour of a computer running the algorithm,
- anticipate the results of small Python programs for the exchange,
- run the programs and compare their predictions.

The main question is to make the students experience the process of writing an algorithm (the abstract part of a program) and play the roles of the different hardware components when running the program (as a physical object).

The theoretical motivation lies in the problems inherent to the definition of the ontological status of the notion of computing and the dual nature of computer programs [1]. The ontological problem is fundamentally based on the question "what is a computer program?". We could defend that a computer program is an abstract entity, a mathematical entity, with independence of its physical implementation, or we could sustain that a program is a concrete entity, a physical entity, this is, the physical machine that actually computes that which the program develops.

The tension between the abstract/concrete duality (mathematical/physical) of programs influences the way that the students think. In [8] the authors addressed the question “What are novice students’ conceptions of computer programming” and found the following categories, among others:

- computer programming as writing text, in which to program is understood as to write a text in a foreign language, the computer’s language,
- computer programming as describing actions, in which there is awareness of the relation between the program text and the actions that take place when the program is executed.

The authors point out that the correspondence between text and action is difficult to capture and programming is regarded as an almost mystical way of thinking. The assignment statement is a simple instruction that clearly reflects this correspondence. Indeed, from the point of view of the program text, the assignment gives a value to a variable (a mathematical variable), and from the point of view of the program as executable object, the assignment alters a memory cell (a physical object). This point is clearly expressed in [4], as the authors write; “... our subjects seem to have guessed,..., that the questions were about change rather than mathematical equality...” (page 11). How to help students in understanding programming variables has largely been a concern [4-7]. Some authors emphasize the physicality of the variables associated with the memory cells [9-11]. In those, roles of variables in program implementation are classified by the relationship between the variable and the value stored during the execution of a program. For example, a "walker" is a variable that is used to traverse a data structure a "follower" is a variable that is used to store the contents of a variable, prior to the execution of a sentence.

The paper is organized as follows; Section 2 describes two activities with concrete material. Section 3, describes the simulation, where the students play roles as computer components. Section 4 describes students' work with Python programs. These activities are described as they were planned, and in Section 5 some results of students' work are analysed. Sections 6, 7 contain acknowledgements and references respectively. The worksheets provided to students are included in Section 8 (Appendix).

2. First part

The first part of the study includes two activities, described below.

2.1 Activity 1

We work with drinking glasses and liquids of different colours. The glasses are divided into 4 identical sets of 8 glasses each, and the students are divided into 4 groups. Each group of students works with a set of glasses. Each group will have 3 to 4 students. The groups work independently.

In the first activity, the groups are shown two glasses, as shown in Figure 1. The task consists in exchanging the liquids without altering or spilling them. That is to say: the red liquid must be poured into glass[2], and the green liquid into glass[1].

Figure 1

The groups are expected not to be able to complete the task. The students are asked why this is the case and what they would need to complete the task. The aim is to raise their awareness about the need to have a third empty glass to exchange the liquids.

2.2 Activity 2

In the second activity, a sequence of glasses is presented (more than two) as shown in Figure 2; some are empty (in white) and others full (coloured). Some are made of glass, others of plastic, and others of acrylic (marked respectively “G”,”P”, and “A” in the figure). The red and green liquids are corrosive, therefore they can only be poured into glasses made of glass. The task is the same as in activity 1: exchange the green and red liquids without mixing them.

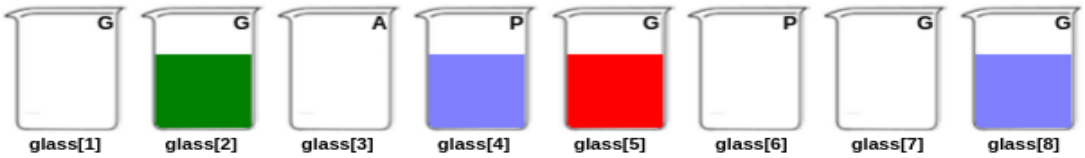


Figure 2 - Initial state of the glasses

In this case, the groups are expected to succeed and to use a third auxiliary glass. Then, the groups are given the following questionnaire. Every member of each group must provide his/her responses in writing, to ensure that all of them have participated in the activity (in the ellipsis, the values are 1 or 7 as the auxiliary glass chosen by the group).

2.2.1 Questionnaire on activity 2

1- Describe how you completed the task, mentioning glasses by their numbers.

2 - Why did you choose glass[…] and not glass [8] or [6], for instance?

3 - Why did you choose glass [...] and not glass [...]?

4 - Draw a sequence of glasses so that the task could not be completed.

2.2.2 Writing the program

We work with the sequence of instructions resulting from the first question of the questionnaire above. We ask students: could we use glass 1 instead of glass 7 and vice versa? (Glass 1 has the same characteristics as glass 7). Is it the same if we begin with glass[5] or glass[2]? The different sequences are written down, starting with glass[5]:

auxiliary glass used is glass[7]	auxiliary glass used is glass[1]
1. pour contents of glass[5] into glass[7]	1. pour contents of glass[5] into glass[1]
2. pour contents of glass[2] into glass[5]	2. pour contents of glass[2] into glass[5]
3. pour contents of glass[7] into glass[2]	3. pour contents of glass[1] into glass[2]

We then explain that the person writing the program (the user) is interested in the final result; exchanging the contents of glasses 2 and 5. The user does not mind whether the computer uses glass1 or glass 7 as the auxiliary glass, moreover, does not know the state of the glasses. Therefore, the user will call this glass “x”. Additionally, to check that the contents have been exchanged, the user needs to verify the initial state and the final state of the glasses. Therefore, the user provides the computer with the following *general program*:

1. show state of glasses
2. pour contents of glass[5] into glass[x]
3. pour contents of glass[2] into glass[5]
4. pour contents of glass[x] into glass[2]
5. show state of glasses

3. Second part

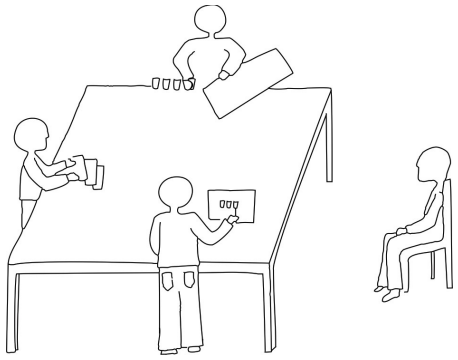
In this second part, our work was inspired by "Computer Science without a computer" (csunplugged.org). Students continue working in groups of 4, where 3 of them will act as the different components of a simplified computer. The fourth student will be **the user.** The user gives the above program to the student acting as the **CPU**. The other students act as the **ALU/Memory** and the **Display.**

The purpose of this simulation is to give the students a small taste of what computers do and about the role played by the different hardware components when running a program, and how they interact [12].

The aim is to highlight the fact that computers simply follow the instructions of a program, which does not mean that they "understand" what they are achieving in the process. This is the first step in a process of awareness students must go through about the importance of writing a program correctly; if the program has a mistake that leads to an unwanted end result, the computer will still execute the instructions as they were written

3.1 Activity

Students are divided as shown in the image.



There is a table, with the row of 8 glasses from Activity 2, and a piece of cardboard to hide or show them. Only the **ALU** has access to this. The **CPU** receives the program and each student receives a worksheet with instructions (see Appendix). The roles can be described as follows:

- The **CPU** (Central Processing Unit) receives the program, reads the instructions one by one and communicates them to the others as applicable. Each time the execution of an instruction is verified, the instruction is ticked off, and the next instruction is executed. This is repeated until all the instructions have been ticked off.
- The **ALU** (Arithmetic Logic Unit) student executes the instructions received from the **CPU**, one by one. The glasses are hidden behind the piece of cardboard. After executing a "show" type instruction, the **ALU** turns the cardboard so that only the **CPU** can see where the glasses are (this should remain hidden from the **Display** and the **User**).
- The **Display** has a sheet with a drawing of the empty glasses, which they modify according to the orders received from the **CPU,** and which is then shown to the **User**.
- The **User** gives the program to the **CPU** and checks whether the exchange was successful or not. This student stands facing the **Display**. This position must be kept at all times, since the **Display** will communicate with the **User** by showing an image.

Students must nod to each other to communicate the completion of the instructions. We have written the program instructions on slips of paper, which are placed on the table so that only the CPU student can see them. For each instruction of the program, CPU choose the slip of paper and pass it on to ALU or to Display. We first work with an example so that students can play their roles using their worksheets and material. The initial state of the glasses is shown in Figure 2.

3.2 Running the program

Once the example has been shown successfully, students take their positions and the **User** gives the program above to the **CPU**.

The **ALU** and the **Display** must always face the **CPU,** who nods to indicate that some action is needed. They must do the same to show they have completed the actions requested.

- The **CPU** reads the first instruction.

    1 . show state of glasses.

- The **CPU** selects the corresponding slip and gives it to the **ALU.**

- The **ALU** shows the **CPU** the state of the glasses by turning the cardboard.

The **CPU** has the following table and fills it in with empty/coloured/full information (full means that the glass contains another liquid).

glass[1] G	empty		
glass[2] G	green		
glass[3] A	empty		
glass[4] P	full		
glass[5] G	red		
glass[6] P	empty		
glass[7] G	empty		
glass[8] G	full		

- The **CPU** looks at the **Display** and shows him/her the table (the **Display** must always face the **CPU**).

The **Display** worksheet states that when the **CPU** shows a table, the **Display** must colour the glasses in the first row of her/his worksheet accordingly and then show it to the user (who is always facing the **Display**).

- The **Display** colours the glasses according to the information on the table and shows them to the **User.** The **Display** nods to notify the **CPU** that the action is complete.

- The **CPU** ticks off the instruction and reads the second instruction:

    2 . pour contents of glass[5] into glass[x]

In this case, the **CPU** looks at the **ALU,** selects the corresponding slip and gives it to the **ALU.** The **ALU** knows, according to his/her worksheet, that he/she must:

- look for an empty glass made of glass

- replace the “x” with the number of the glass chosen *in all the slips where “x” occurs*
- pour the contents of glass[5] into it

- The **ALU** performs this action hidden from the others, but tells the **CPU** the action is complete (nodding).

- The **CPU** ticks off the instruction and reads the next instruction:

3. pour contents of glass[2] into glass[5]

selects the corresponding slip and gives it to the **ALU.**

- The **ALU** pours the contents of glass [2] into glass[5], and nods to indicate the action is complete.

- The **CPU** ticks off the instruction and reads the next one:

4 . pour contents of glass[x] into glass[2]

selects the corresponding slip and gives it to the **ALU.**

- The **ALU** replaces the “x” with the number of the glass previously selected and pours its contents into glass[2]. Then the **ALU** nods to notify the **CPU** that the action is complete (nodding).

- The **CPU** ticks off the instruction and reads the next one:

5. show state of glasses

This time a new column in the table is completed by the **CPU**, shown to the **Display**, who in turn colours the glasses of the second row of her/his worksheet and shows the picture to the **User**.

When the **Display** indicates the action is complete, the **CPU** ticks off the last instruction and the program ends.

The **User** should see, at the end, the two drawings of the initial state and the final state as in Figure 3 (see Appendix).

4. Third part

The final activity consists of asking the students to anticipate the results of small Python programs (the students have been instructed in basic Python). They are then asked to run them and compare those results with the predictions they made. The students worked individually and wrote their answers down. The activity is described as follows.

A variable **v** is assigned the value ['w', 'w', 'g', 'w', 'w', 'r', 'w', 'w'], w, r, g for white, red and green respectively. Recall that v[i] is the value in position i, for $0 \leq i \leq 7$ (for example, v[2] is 'g').

For each program in the table below, answer the questions.

Case 1	Case 2	Case 3	Questions
print v v[0] = v[2] v[2] = v[5] v[5] = v[0]	v[0] = v[2] v[2] = v[5] v[5] = v[0] print v	print v v[0] = v[2] v[2] = v[5] v[5] = v[0] print v	1. Which is the displayed result of each program when executed? 2. Which is the value of v[0] before and after execution? 3. Does the computer interchange v[2] and v[5] in every case? Why or why not?

5. Preliminary analysis

The aim of this research is on the one hand, to make the students experience the process of writing an algorithm (the abstract part of a program) and play the roles of the different hardware components when running the program (as a physical object). On the other hand, to gather information about the preconceptions of students regarding the process of *a computer* running a program. Finally, to offer new insights about how student's preconceptions relate to their experience of performing the actions themselves. Since the study described in this article was performed very recently, I am offering the following conclusions as a preliminary discussion around the facts found so far. An in-depth analysis of the results will be carried out as part of further research.

Thirty-seven works of the students were analysed. Regarding the last activity (Third part, Section 4), just two students answered question 1 as expected, pointing out the value of the variable **v** displayed in each print sentence, as shown in the table below:

Case 1	Case 2	Case 3
['w', 'w', 'g', 'w', 'w', 'r', 'w', 'w']	['g', 'w', 'r', 'w', 'w', 'g', 'w', 'w']	['w', 'w', 'g', 'w', 'w', 'r', 'w', 'w'] ['g', 'w', 'r', 'w', 'w', 'g', 'w', 'w']

Most of students wrote different values of the variables v[0], v[2] and v[5]. Examples:

in all the cases	v[0] = 'g'	v[2] = 'w'	v[5]= 'r'

Case 1	Case 2	Case 3
v[0] = 'r'	v[0] = 'g'	v[0] = 'w'
v[2] = 'g'	v[2] = 'w'	v[2] = 'r'
v[5] = 'w'	v[5] = 'r'	v[5] = 'w'

There are some limitations to the possibilities to draw conclusions about questions 1 and 2, due to the variety of responses. Instead, these give rise to further questions: Which is the print sentence that students consider? Is the assignment interpreted in reverse for some sentences? Is v[2] the auxiliary variable? Why? Which are the points of the activities of First and Second part (successfully done) that can be improved to impact in Third part? How can the questions of Third part be better formulated? How do students interpret the differences between their predictions and the results when running the programs? These points will be included in future activities of the project.

Regarding question 3, it is quite clear that more or less half of students believe that the interchange does not take place without print sentences. Indeed, seven students answered that without “print” sentences, the exchange is not performed, and eleven students believe that an error should occur (and expressed orally that this is because there is no print). This result is, to some extent, coherent with the simulation (Second part), where students tended to show (print) the glasses (turning the cardboard), *after each of the sentences of the exchange* (as if only showing the exchange, it takes effect).

Regarding activities 1 and 2 of First part (Section 2), although both were successfully done, there is a significant gap between the actions students are able to perform and their ability to answer questions that require thinking or imagining. For example, in question 1 of the questionnaire on Activity 2 (Section 2.2.1), where they were asked to write step by step the exchange just made, some students needed to perform the action again to be able to write it down. Likewise, in question 4 many students show difficulties to draw a sequence for which it is not possible to make the exchange: either they did not understand the question or they did not know how to provide an answer, despite having responded immediately and without doubt why the activity 1 (Section 2.1) could not be performed.

At the end of the activities, it was stressed that the simulation reflects the behaviour of the computer in a simplified form and is, in some cases, inaccurate. However, others are properly simulated; an algorithm is transformed into internal codes, the instructions alter physical objects, and the result is displayed to the user in a format that is understandable to human beings. The cells of the memory can be available for program instructions or not, as well.

Finally, I argue that, although the activities were conducted for a research study, I find that it is possible and beneficial to conduct similar exercises as learning activities. Students were highly motivated and enthusiastic which was documented in video recordings and photographs.

6. Acknowledgements

Thank you to Néstor Larroca and Patricia Añón for helping in doing the activities in their classes. The comments of the referees are gratefully acknowledged.

7. References

[1] da Rosa, S., Chmiel A., Gómez F. (2015) Philosophy of Computer Science and its Effect on Education -Towards the Construction of an Interdisciplinary Group, Clei Electronical Journal, Volume 19: Number 1: Paper 5. [Online]. Available: http://www.clei.org/cleiej/volume.php

[2] Peyton Jones, S. (2013) Computing at school in the UK. [Online]. Available: http://research.microsoft.com/en-us/um/people/simonpj/papers/cas/

[3] Dowek, G. (2005). Quelle informatique enseigner au lycée? http://lsv.fr/~dowek/Enseignement/lycee.html

[4] Dehnadi, S., Bornat, R., The camel has two humps School of Computing, Middlesex University. [Online]. Available: http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

[5] Wagner, S. (1983) What are these things called variables? Mathematics Teacher, pp. 474–479.

[6] Samurçay, R. (1989) The concept of variable in programming: Its meaning and use in problem-solving by novice programmers, in Studying the Novice Programmer, E. Soloway and J. C. Spohrer, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers,, ch. 8.

[7] Du Boulay, B. and Benedict J. (1986) Some difficulties of learning to program Journal of Educational Computing Research, 2(1), 57-73.

[8] Thuné, M., Eckerdal, A., (2009) Students' Conceptions of Computer Programming (excerpt from Variation theory applied to students' conceptions of computer programming. European Journal of Engineering Education, 34(4), 339−347)

[9] Byckling P., Sajaniemi, J., Roles of variables and programming skills improvement, in SIGCSE, D. Baldwin, P. T. Tymann, S. M. Haller, and I. Russell, Eds. ACM, 2006, pp. 413–417. [Online]. Available: http://doi.acm.org/10.1145/1121341.1121470

[10] Nikula, U, Sajaniemi, J., Tedre, M. and Wray, S. (2007) Python and roles of variables in introductory programming: Experiences from three educational institutions, JITE, vol. 6, [Online]. Available: http://www.jite.org/documents/Vol6/ JITEv6p199 214Nikula269.pdf

[11] Sorva, J., Karavirta, V, Korhonen, A. (2007) Roles of variables in teaching, JITE, vol. 6, pp. 407–423. [Online]. Available: http://www.jite.org/documents/Vol6/JITEv6p407423Sorva280.pdf

[12] Gary Kacmarcik (2010) How computers work. [Online]. Available: http://cse4k12.org/how_computers_work/index.html

8. Appendix

Worksheets

8.1 CPU

Your job as the Central Processing Unit (CPU) is to execute the program. This means reading each instruction, choosing, from the slips available, one that matches the instruction, and sending the slip to the "ALU/Memory" and "Display" classmates as follows:

- the ALU/Memory should be given instructions regarding
    - information about the state of the glasses
    - changing the contents of the glasses
- the Display should be given instructions about showing the user the drawing of the state of the glasses.

The ALU and the Display will be looking at you, awaiting orders, and they will let you know that the action is complete by nodding. You should do the same when you have finished. The table is to be completed with the state of the glasses when the **ALU** shows them to you. For example, if the instructions are:

show state of glasses
pour contents of glass[5] into glass[x]

Read the first instruction, choose the corresponding slip and give it to the **ALU**. The **ALU** turns the cardboard, then you must look at the glasses, complete the table and notify the **ALU** that he/she can return the cardboard to its original position by nodding.

glass[1] G	empty		
glass[2] G	green		
glass[3] A	empty		
glass[4] P	full		
glass[5] G	red		
glass[6] P	empty		
glass[7] G	empty		
glass[8] G	full		

Turn to the **Display** and show him/her the table (the **Display** must always look at the last column of the state of the glasses). When the Display nods indicating that the instruction has been executed, tick off the instruction and read the next one. To execute it, nod to the **ALU** and pass on the slip with the next instruction: glass[5] to glass[x]. When the **ALU** nods, tick off instruction 2 and read the next one. This is repeated until you have ticked off the last instruction.

8.2 ALU/Memory

Your job as the ALU/Memory is to execute the instructions given by the CPU about the glasses on the table. You must look at the CPU all the time, awaiting instructions. If the CPU gives you a slip with an instruction on it, you must:

- If you read "glass[n] to glass[x]" (n between 1 and 8), find the suitable glass (empty and made of glass), and when you find it, cross out the "x" and write the number of that glass (the same

glass must be used for all the instructions that include “x”). Pour the contents of glass[n] into the glass you have chosen as glass “x”.

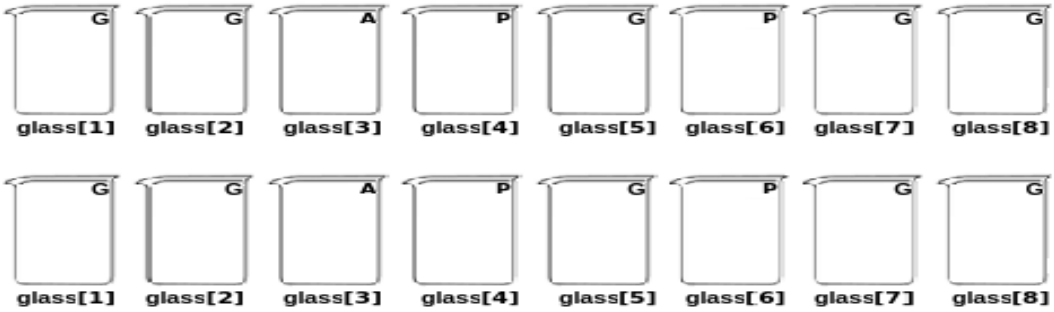
- If you read “glass[n] to glass[m]” (n and m are numbers between 1 and 8), pour the liquid from glass[n] into glass[m].

Every time you complete an instruction of this kind, signal this to the CPU by nodding.

If the CPU gives you a slip with a “show” type instruction, turn the cardboard and let CPU see the state of the glasses. When the CPU signals that he/she has seen it by nodding, return the cardboard to its original position.

8.3 Display and User

Your job is to wait until the CPU tells you to what to show the User. You must look at the CPU all the time, awaiting instructions. You receive the following drawings:



When the CPU shows you a table, you must colour an image, starting with the first one, according to the information on the table (colours will always be the ones in the last column, that is to say, the one on the far right). Then, show it to the User and nod to the CPU indicating you have finished.

The **User** should see, at the end, the two drawings: the initial state and the final state.

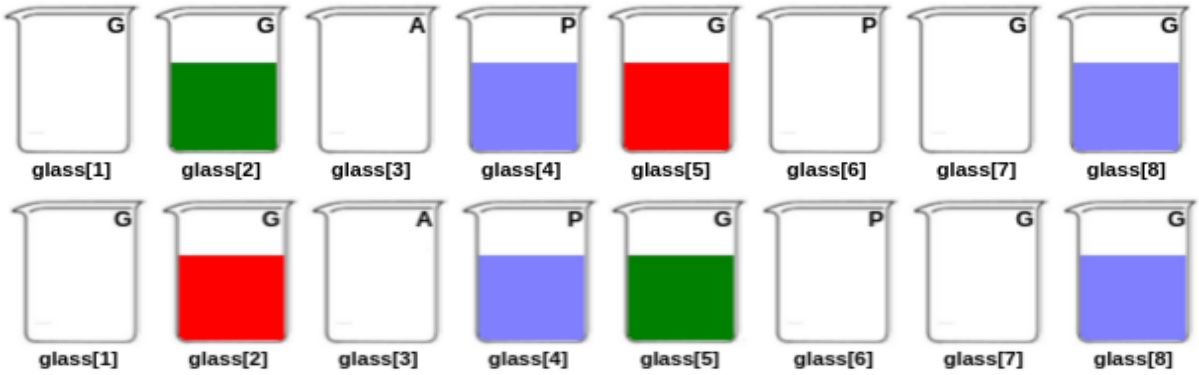


Figure 3 - Initial and final state of the glasses