Info 3

Laboratory 6

18.06.2020

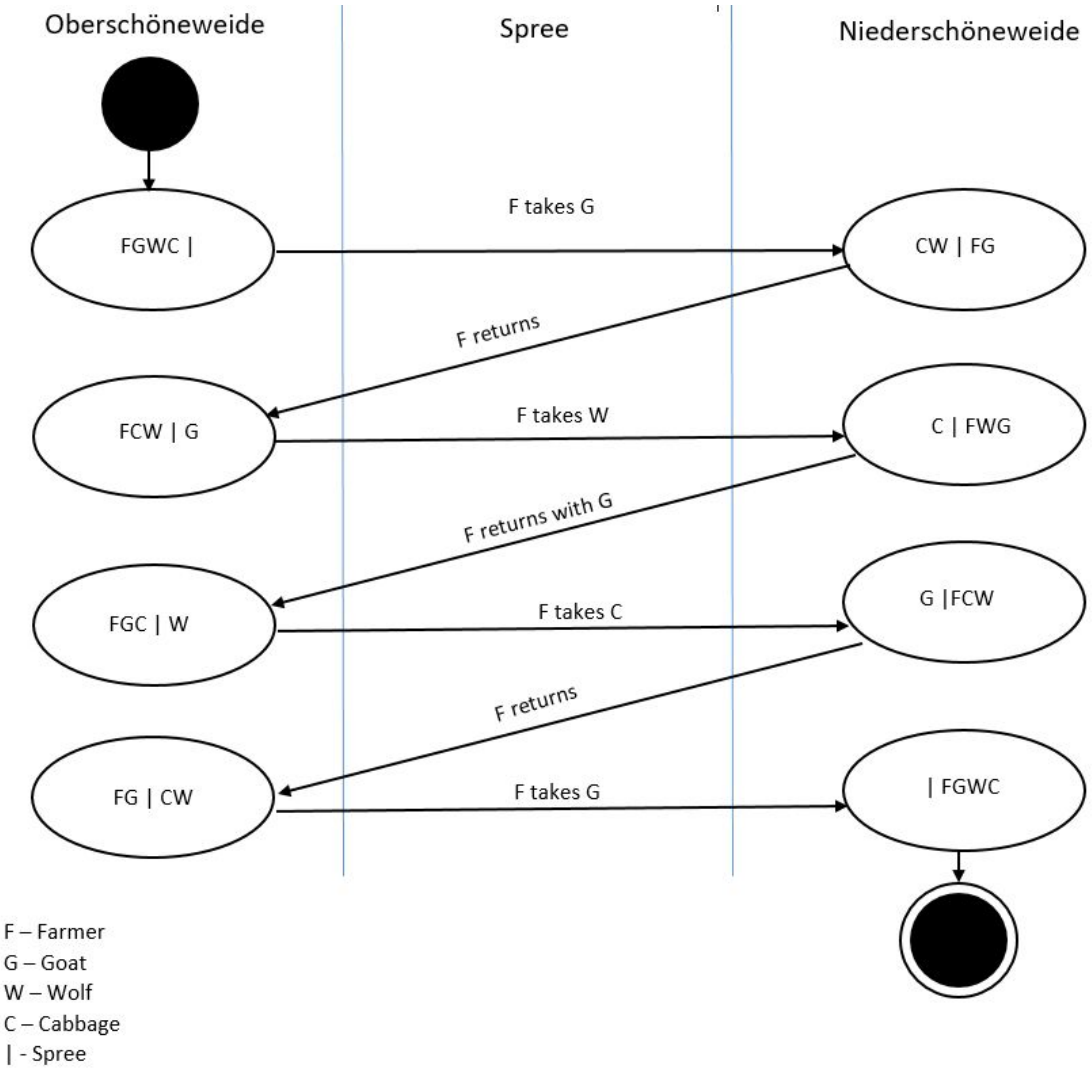
Lab 6: State-Transition Diagrams

Niklas Lengert s0563290
Pavel Tsvyatkov s0559632
Robin Jaspers s0568739
Nataliia Azarnykh s0568691

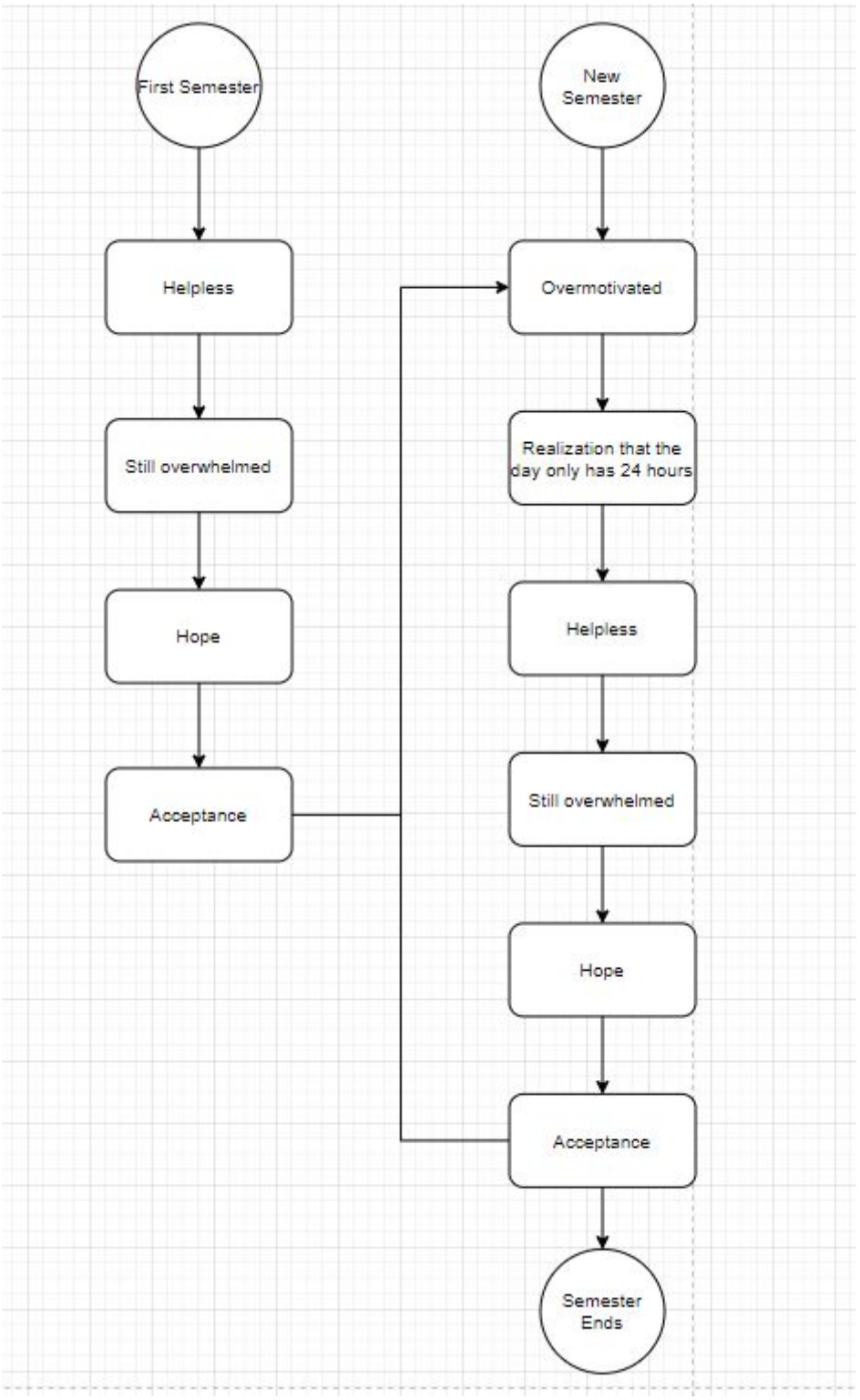
TASKS:

1. **Model the modified goat/cabbage/wolf problem: The farmer is in Niederschöneweide and wants to get his goat, his cabbage and his wolf over to Oberschöneweide. Only one thing can fit in his boat at a time beside himself. He cannot leave the cabbage and the goat or the goat and the wolf alone on the same side of the river, for obvious reasons. Is it possible for him to get all three possessions across the Spree? Draw a State Transition Diagram modelling a solution to this problem.**

(We drew a sketch to get an idea about how we want to make the diagram.
We also found a drawing on this from Informatics 2 - see **Appendix** in the end)



2. **Model the states an IMI student passes through from the first until the sixth semester. (Special prize for the most humorous model that is not offensive.)**



3. **Model an algorithm that determines if a given string is a proper floating-point number, i.e. [sign] integerpart dot fractional part [E exponent].**

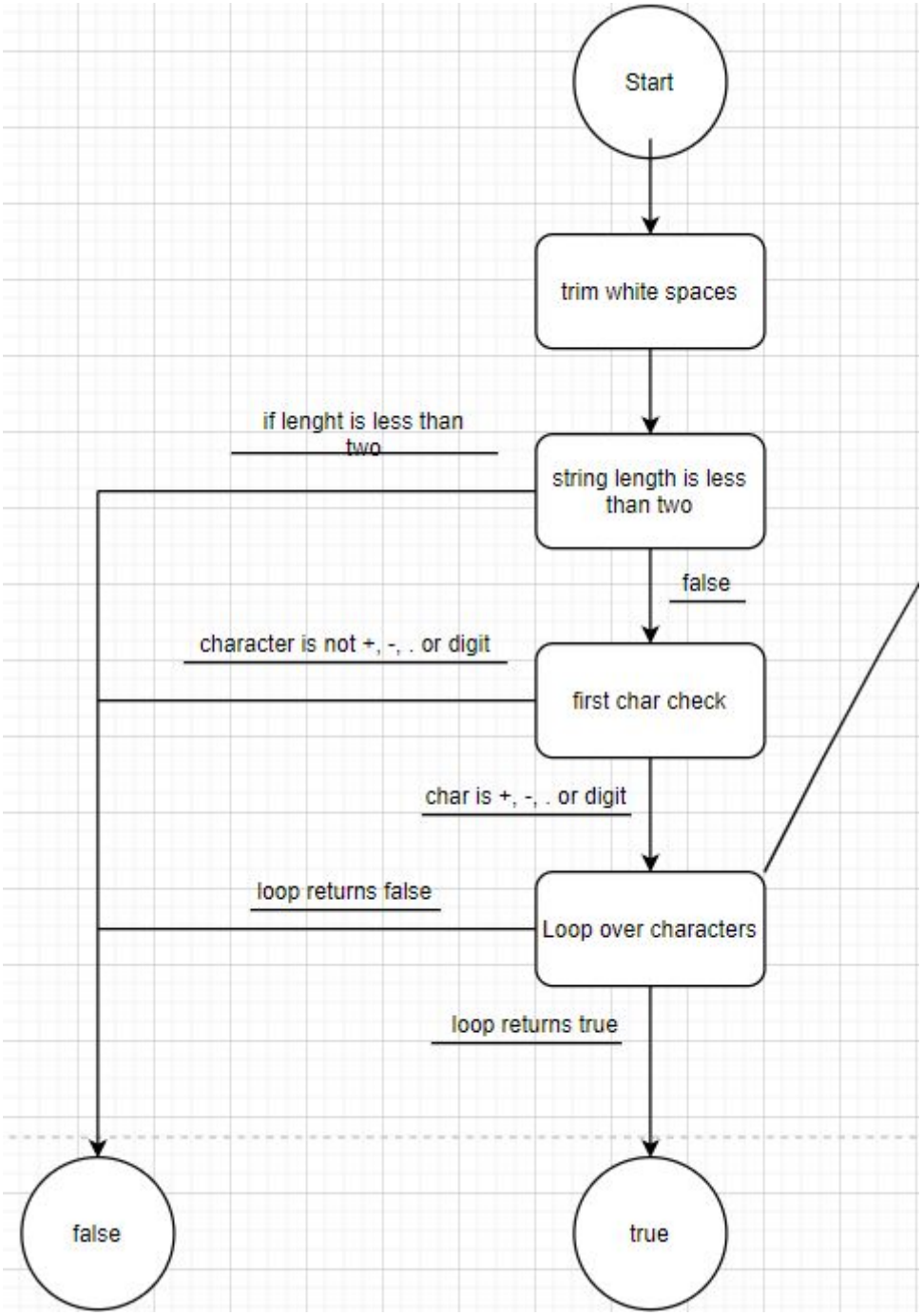
Since we kind of rushed this one the first time we thought about it, we overlooked some important things we had to pay attention to. Then we decided to make a list of some condition checks. In that way we would be able to keep track of things that are or aren't allowed.

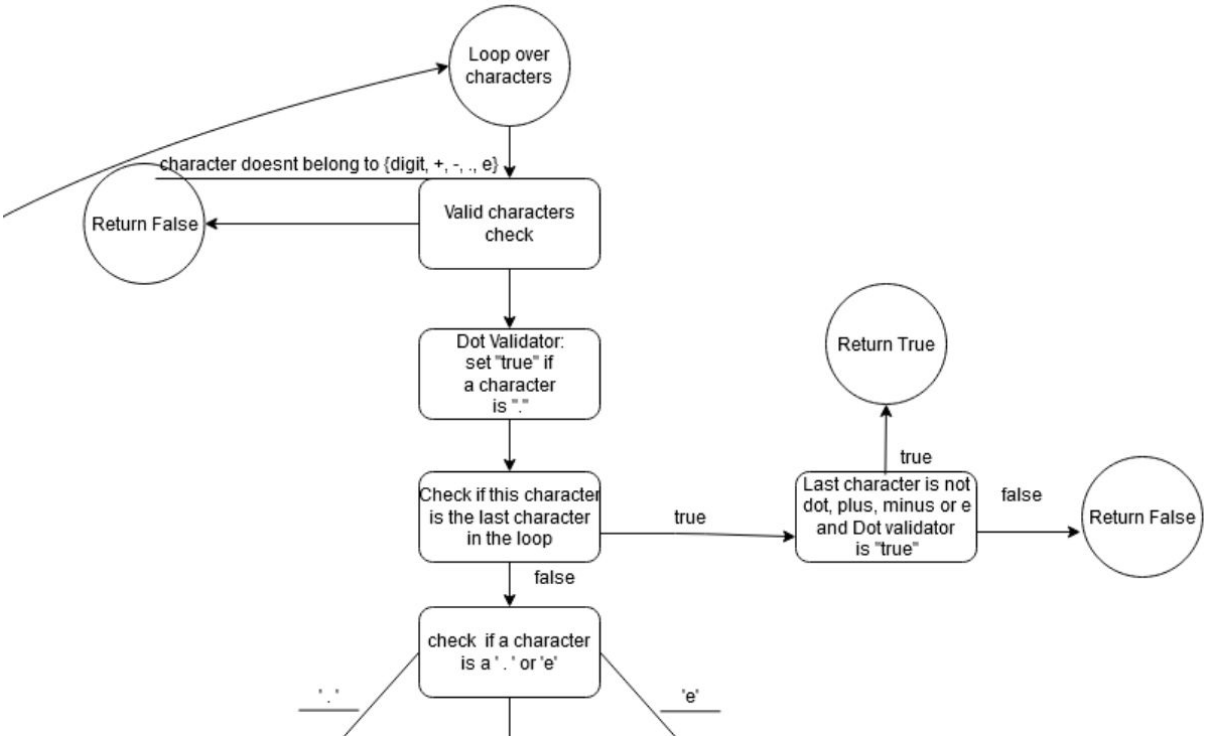
https://www.geeksforgeeks.org/check-given-string-valid-number-integer-floating-point/
Here, we found some good check conditions, but we also thought of many additional ones while discussing.

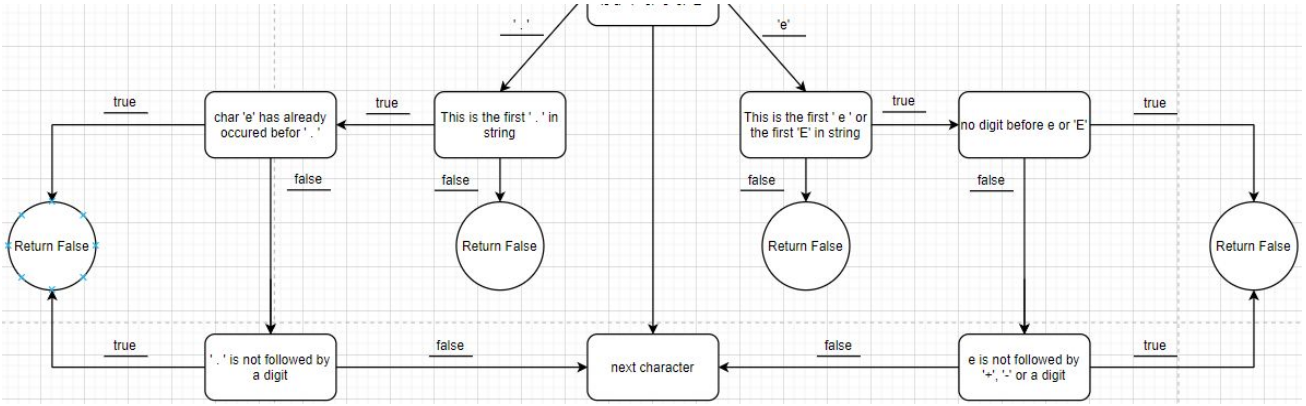
1. Trim whitespaces
2. Trim any leading zeros, i.e 0001.234
3. Make sure the characters in the string belong to {+, -, ., e, E, numbers from 0 to 9 }
4. + and - can occur either at the beginning or after 'e' , 'E'
5. The string ends with a digit (or doesn't end with a +, - , dot, 'e' or 'E'
6. Since we check if it's a floating-point number, we need to have a dot showing up somewhere in the string
7. There shouldn't be more than one dot in the string
8. The dot is followed by a digit
9. There is no dot after 'e' or 'E'
10. There should be no more than one of either an 'e' or an 'E' character in the string
11. 'e' or 'E' cannot occur before the dot
12. 'e' or 'E' is followed by a sign, either + or -, or a digit

Then we assume that a float could be represented in a few different ways, i.e: 0.7, -0.7, 0.77, 0.77e1, 0.77e+1, 0.77e-1

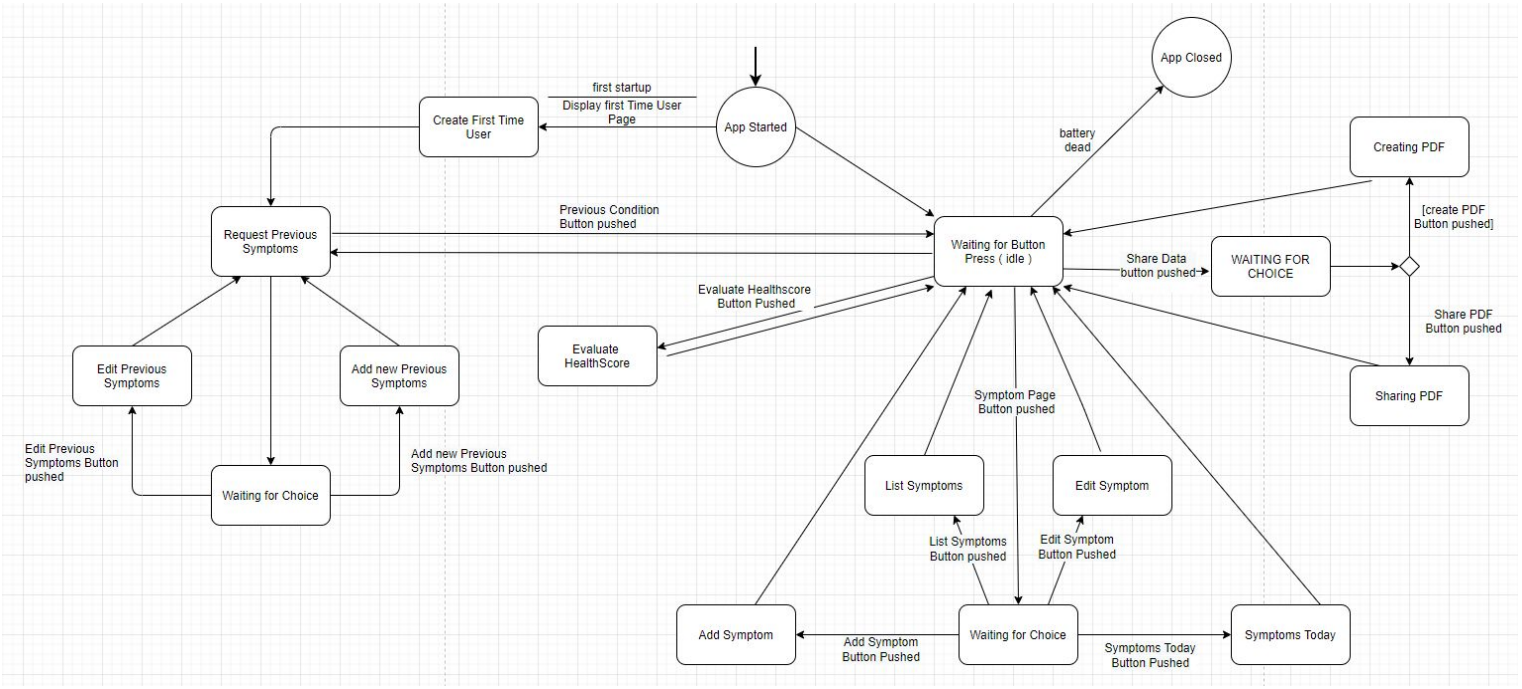
The model of the algorithm got way too big so we had to show the pictures below separately. Alternatively, here's a link to our schema in google drive: **https://drive.google.com/file/d/1JQ5A4gZDKRrGs5lkiYEA_kny0srwCUHu/view?usp=sharing**


Start
trim white spaces
if lenght is less than two
string length is less than two
false
character is not +, -, . or digit
first char check
char is +, -, . or digit
loop returns false
Loop over characters
loop returns true
false
true


Loop over characters
character doesnt belong to {digit, +, -, ., e}
Return False
Valid characters check
Dot Validator: set "true" if a character is "."
Return True
true
Check if this character is the last character in the loop
true
Last character is not dot, plus, minus or e and Dot validator is "true"
false
Return False
false
check if a character is a ' . ' or 'e'
' . '
'e'


' . '
'e'
true
char 'e' has already occured befor ' . '
true
This is the first ' . ' in string
false
false
Return False
Return False
true
This is the first ' e ' or the first 'E' in string
true
no digit before e or 'E'
true
false
false
Return False
Return False
Return False
true
' . ' is not followed by a digit
false
next character
false
e is not followed by '+', '-' or a digit
true

4. Model the states for your Corona App.



Reflection

Niklas: Compared to the last several exercises this was a lot easier to grasp because over the course of the last weeks I sometime did not understand what to do at first glance but this lab was very clear and it was also a lot of fun to create some of the diagrams. Thinking about algorithms is always fun for someone like me who loves math.

Pavel: The first two tasks were not difficult, we actually enjoyed coming up with ideas about how to draw the second diagram. We had to discuss a lot about the third task and consider many things while drawing the diagram. We were coming up with new condition checks all the time, because there were many things that had to be fulfilled to check if a string is a floating point. I think we were able to discuss well how we wanted to do the states for our app. We had a lot of back and forth changing, but we managed to draw it in the way we thought it should work. It was challenging, because it was the first time drawing state transition diagrams, but it was a great learning experience.

Robin: For this exercise our first two tasks were a little bit different to get into the topic. I enjoyed that they were quite playful which is a good way to get into it. At the point finishing them i was experienced enough to get into harder stuff. Most the time of task 3 i had to think about the problem rather than the State-Transition Diagram. Getting in the last task wasn't that hard since i had a rough idea about how to do it. There was still much to improve though.

Nataliia:

Challenging, but very interesting exercises. I especially love the first one and the third one. Algorithms are always fun. We described states diagram in all exercise, except exercise 3. In exercise 3 we had to show an algorithm as a diagram. It made me reflect on the differences between STD and algorithm diagram. The difference is huge.

Time spent:

Reading both chapters	**30 min**
Exercise 1	**30 min**
Exercise 2	**30 min**
Exercise 3	**2 hours**
Exercise 4	**2 hours**