I started off with figuring out the source-code that was handed out and how the methods use the self-instance. I figured out that the pseudocode used the methods already present and started by filling the function out with those methods. Since the instance the code operates on is an instance of the class “CSP”, I could leave the parameter csp out of the function since it is an implicit parameter.
Since the other methods also use the self-instance, I could simply call the methods directly on the instance. The important part was to remember to add the parameters of the different methods.
To add the different values from the constraint satisfaction problem, I assigned the values of the self-instance to the list “assignment[unassigned_variables]”. This works because the domains, neighbours and constraints are all passed into the self-instance by the other methods. Lastly, I save the self-instance in the “result”-variable. The methods deplete the list of unassigned values and returns the final result.

For the map, I manually typed the names of the countries, entered a new color to the values-list and added the countries to the variables-list. I add the countries to the domain-dictionary and assigns each country the range of colors to select from. I enter the neighbours and assign each country the constraint-function.

Now when I run the project, each country is assigned a color automatically by the recursive-backtracking function.