

TITLE

CSP. ECLiPSe CLP

OBJECTIVES

- Solving Constraint Satisfaction Problems using Constraint Logic Programming (CLP)

PREREQUISITES

- [ECLiPSe - Download](#)
- Course 7

RESOURCES

- [ECLiPSe - A Tutorial Introduction](#) PDF
- [ECLiPSe - A Tutorial Introduction](#) HTML

LAB

Configure ECLiPSe CLP

1. Using the download link in the Prerequisites section download the suitable version of ECLiPSe for your operating system.
2. The installation should be straight-forward (Next...Finish).
3. Well done! You have successfully installed the tool.

Write and Run your first CLP program (You can use the .pl file provided in Classroom [send_more_money.pl])

1. Create a new .pl or .ecl file in which you will write your code.
2. Implement your code in this file using a text editor of your choice.
3. Now, for running your program you have two options:

a. Command Line Interface (CLI)

```
PS C:\Users\Ionut\Facultate\Laboratoare\Artificial Intelligence\2019-2020> eclipse
ECLiPSe Constraint Logic Programming System [kernel threads]
Kernel and basic libraries copyright Cisco Systems, Inc.
and subject to the Cisco-style Mozilla Public Licence 1.1
(see legal/cmpl.txt or http://eclipseclp.org/licence)
Source available at www.sourceforge.org/projects/eclipse-clp
GMP library copyright Free Software Foundation, see legal/lgpl.txt
For other libraries see their individual copyright notices
Version 7.0 #54 (x86_64_nt) Wed Feb 26 22:13 2020
[eclipse 1]: [send_more_money].
source_processor.eco loaded in 0.00 seconds
ash.eco loaded in 0.00 seconds
compiler_common.eco loaded in 0.02 seconds
compiler_normalise.eco loaded in 0.02 seconds
compiler_map.eco loaded in 0.00 seconds
compiler_analysis.eco loaded in 0.00 seconds
compiler_peekhole.eco loaded in 0.02 seconds
compiler_codegen.eco loaded in 0.00 seconds
compiler_varclass.eco loaded in 0.00 seconds
compiler_indexing.eco loaded in 0.00 seconds
compiler_regassign.eco loaded in 0.00 seconds
asm.eco loaded in 0.02 seconds
module_options.eco loaded in 0.00 seconds
ecl_compiler.eco loaded in 0.06 seconds
ic_kernel.eco loaded in 0.00 seconds
lists.eco loaded in 0.02 seconds
linearize.eco loaded in 0.02 seconds
ic_constraints.eco loaded in 0.02 seconds
ic.eco loaded in 0.00 seconds
ic_generic_interface.eco loaded in 0.00 seconds
ic_search.eco loaded in 0.02 seconds
ic.eco loaded in 0.03 seconds
send_more_money.pl compiled 4752 bytes in 0.03 seconds
yes (0.09s cpu)
[eclipse 2]: send_more_money(D).
D = [9, 5, 6, 7, 1, 0, 8, 2]
Yes (0.00s cpu, solution 1, maybe more) ? ;
No (0.00s cpu)
[eclipse 3]:
```

— open ECLiPSe

Loading the file using this syntax: [filename]. or compile(filename).

Compilation info

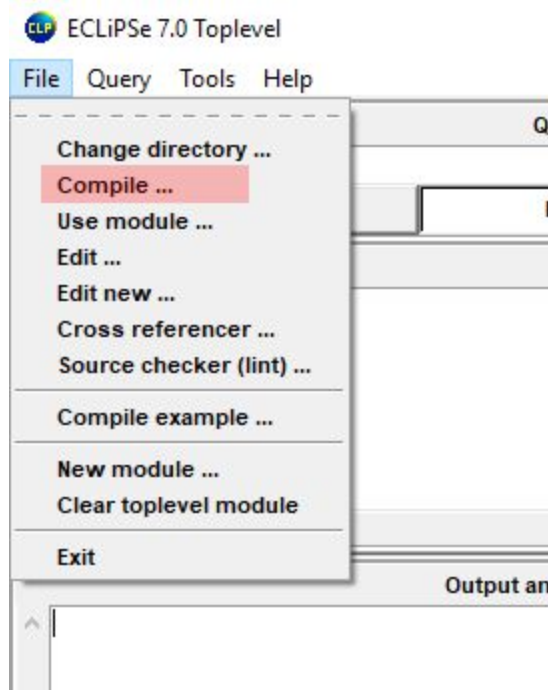
Calling the predicate.

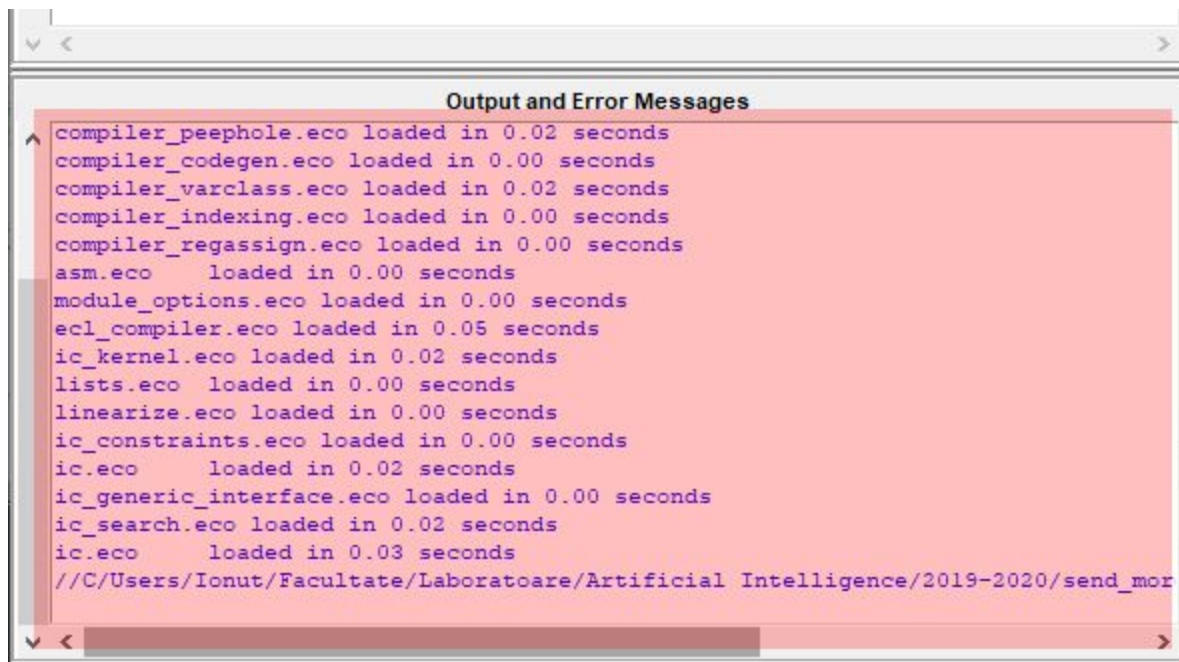
The result.

[Read more here.](#)

b. TkEclipse (GUI)

Open the file: File -> Compile

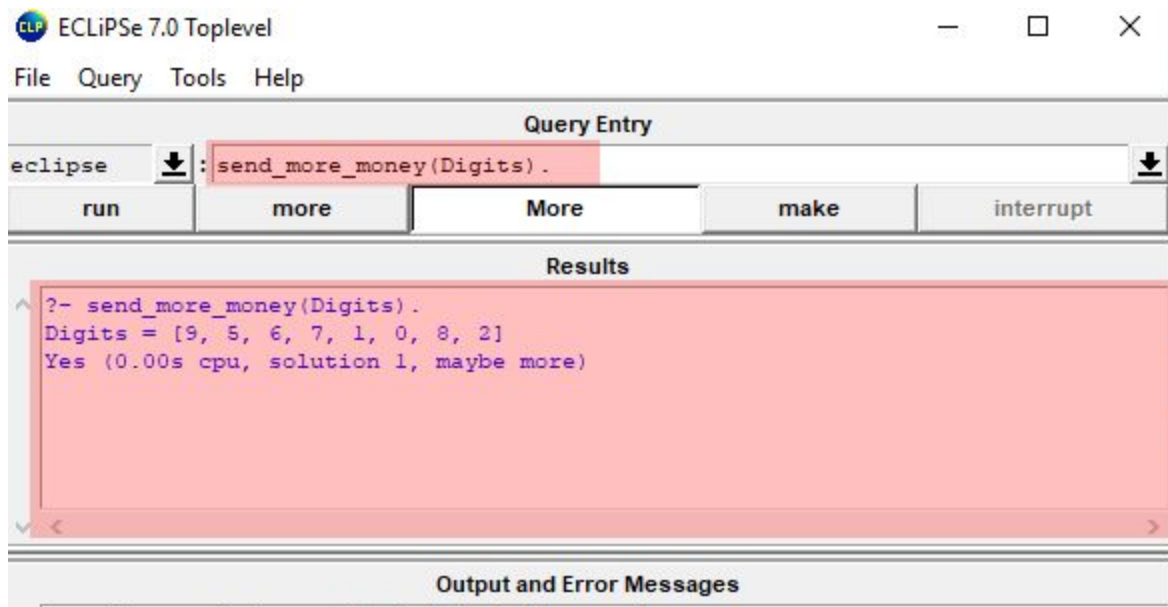




```
Output and Error Messages
compiler_peephole.eco loaded in 0.02 seconds
compiler_codegen.eco loaded in 0.00 seconds
compiler_varclass.eco loaded in 0.02 seconds
compiler_indexing.eco loaded in 0.00 seconds
compiler_regassign.eco loaded in 0.00 seconds
asm.eco loaded in 0.00 seconds
module_options.eco loaded in 0.00 seconds
ecl_compiler.eco loaded in 0.05 seconds
ic_kernel.eco loaded in 0.02 seconds
lists.eco loaded in 0.00 seconds
linearize.eco loaded in 0.00 seconds
ic_constraints.eco loaded in 0.00 seconds
ic.eco loaded in 0.02 seconds
ic_generic_interface.eco loaded in 0.00 seconds
ic_search.eco loaded in 0.02 seconds
ic.eco loaded in 0.03 seconds
//C/Users/Ionut/Facultate/Laboratoare/Artificial Intelligence/2019-2020/send_more
```

If any error (errors will be displayed in red) in the output means that file was loaded and the code was successfully compiled.

Next, you have to run a new query. Write your predicate in the query input and click the “Run” button.



```
ECLiPSe 7.0 Toplevel
File Query Tools Help

Query Entry
eclipse : send_more_money(Digits).
run more More make interrupt

Results
?- send_more_money(Digits).
Digits = [9, 5, 6, 7, 1, 0, 8, 2]
Yes (0.00s cpu, solution 1, maybe more)

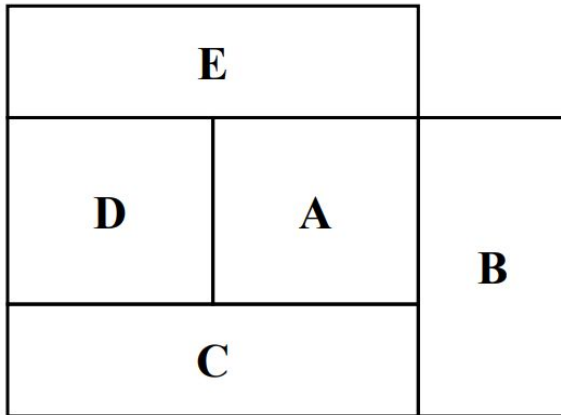
Output and Error Messages
```

As you can see the result is then displayed in the “Results” panel.

Please analyze the provided code for the SEND+MORE = MONEY cryptarithmic problem and try to solve as many exercises as you can.

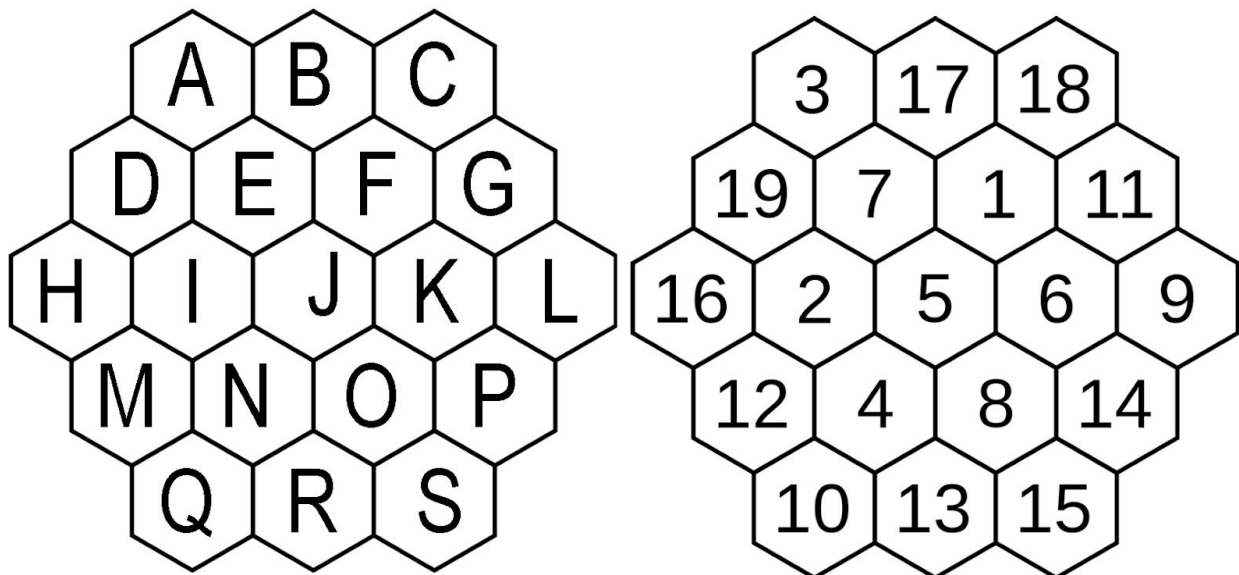
Exercises:

1. Implement $TWO + TWO = FOUR$ example in Course 7 example in ECLiPSe. This problem is similar to the $SEND + MORE = MONEY$ problem.
2. Implement Map Coloring problem in ECLiPSe.



You have a map with 5 surfaces and 4 colors available. You have to write a CLP program that will help us to color the figure so that we will not have two neighbor surfaces colored with the same color. HINT: map your colors with integers.

3. A magic hexagon consists of the number 1 to 19 arranged in a hexagonal pattern:



We have a constraint that all diagonals sum to 38. That is, $A+B+C = D+E+F+G = \dots = Q+R+S = 38$, $A+D+H = B+E+I+M = \dots = L+P+S = 38$, $C+G+L = B+F+K+P = \dots = H+M+Q = 38$.