Diabolic Magic Squares

Arturo Chinchilla Sánchez, Malcolm Davis Steele

April 16, 2016

Abstract

"Logic programming and one of its representatives, PROLOG, make a declarative approach to writing computer programs. Logic programs can be understood and studied using two abstract concepts: truth and logical deduction. One can ask whether an axiom in a program is true, under some interpretation of the program symbols, or whether a logical statement is a consequence of the program. These questions can be answered independently of any concrete execution mechanism.

On the contrary, Prolog is a programming language with precise operational meaning that borrows its basic concepts from logic programming. The Prolog programs are instructions for execution on a computer. These instructions can almost always be read as logical statements and, most important, the result of a computation of a Prolog program is a logical consequence of the axioms in it.

Note, that effective Prolog programming requires an understanding of the theory of logic programming".[1]

Contents

List of Figures

1 Introduction

"A magic square consists of a series of numbers so arranged in a square, that the sum of each row and column and of both the corner diagonals shall be the same amount which may be termed the summation (S). Any square arrangement of numbers that fulfills these conditions may properly be called a magic square." [2]

There are several programming paradigms, such as imperative, functional, logic, and others. For this project the logic paradigm will be used. The Logic Paradigm focuses on logic predicates, in which the most important concept is a relation.

"Instead of specifying instructions on a computer, the logic programming paradigm enables the expression of logic. It is therefore useful for dealing with problems where it is not obvious what the functions should be. In this paradigm, programmers specify a set of facts such as statements or relationships that are held to be true, and a set of axioms or rules (i.e., if A is true, then B is true), and use queries to the execution environment to see whether certain relationships hold and to determine the answer by logical inference. This paradigm is popular for database interfaces, expert systems, and mathematical theorem provers"[5].

For this project is created a program composed by two modules, the interface (Frontend) and the logical part (Backend). The Frontend is in charge to interact with the user, gets the user entries and show the information that the user ask. Also handles the communication with the Backend. The logical part is in charge to calculate the user operations, and return to the Frontend the answer.

2 Problem Description

In this project is proposed to create a program that verifies the user entries and generate diabolic magic squares, using the logic paradigm programming and Prolog like programming language. The implementation of this project, is necessary to create 2 programs, that will be interacting with each other.

- The first part is a Frontend, which purpose will be to interact with the user, take care of the Inputs and Outputs operations and communicates with the second part, the backend.
- The Backend, contain all the logic part, it's in charge of all the diabolic magic squares resolution.

Is allowed develop the Frontend in any language, but the Backend should be developed using Prolog for LINUX. The Backend receive from the Frontend all the request user operations, calculate this, and send the answer again to the Frontend to be displayed to the user.

2.1 What's a "Diabolic Magic Square"?

"The Diabolic Magic Square also, known as the Pandiagonal Magic Square, pandiagonal magic square, diabolic square, diabolical square is the one that satisfies the condition that the square should be magic along the broken diagonals as well as along the two ordinary diagonals"[3].
"In other words, if a Diabolic Magic Square is cut into two pieces along a line between any two rows or any two columns, and the two pieces are interchanged, the real square so formed will also be pandiagonally magic"[3]. See the Figure 1.

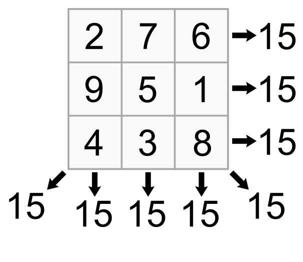


Figure 1: Diabolic Magic Square 3x3.

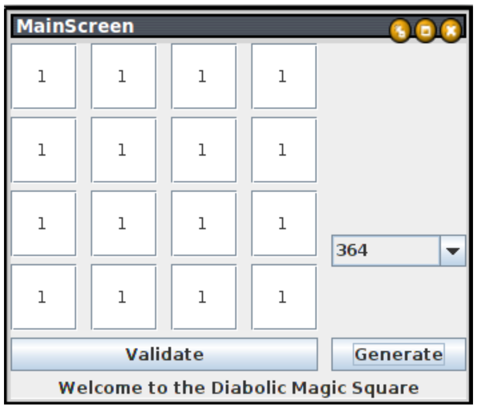
3 User Guide

3.1 hardware and software requirements

- HDD: At least 100MB free.
- RAM Memory: 2GB.
- Processor: Dual Core Processor (1.4GHz)
- OS: Linux Mint 17.2 or another version.
- IDE: IntelliJ IDEA
- Libraries: SWI-Prolog and Java JRE 8.

3.2 How use the program

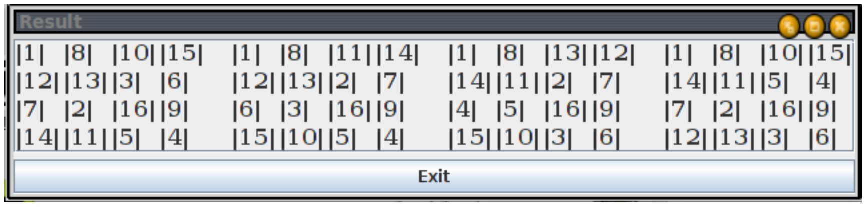
- Open the project with IntelliJ IDEA and run it.
- When you run the program, a window is displayed, and you have two options, entry your Diabolic Magic Square for the program verifies if is or not, or choose generate with the amount of squares.See the Figure 2



Welcome to the Diabolic Magic Square

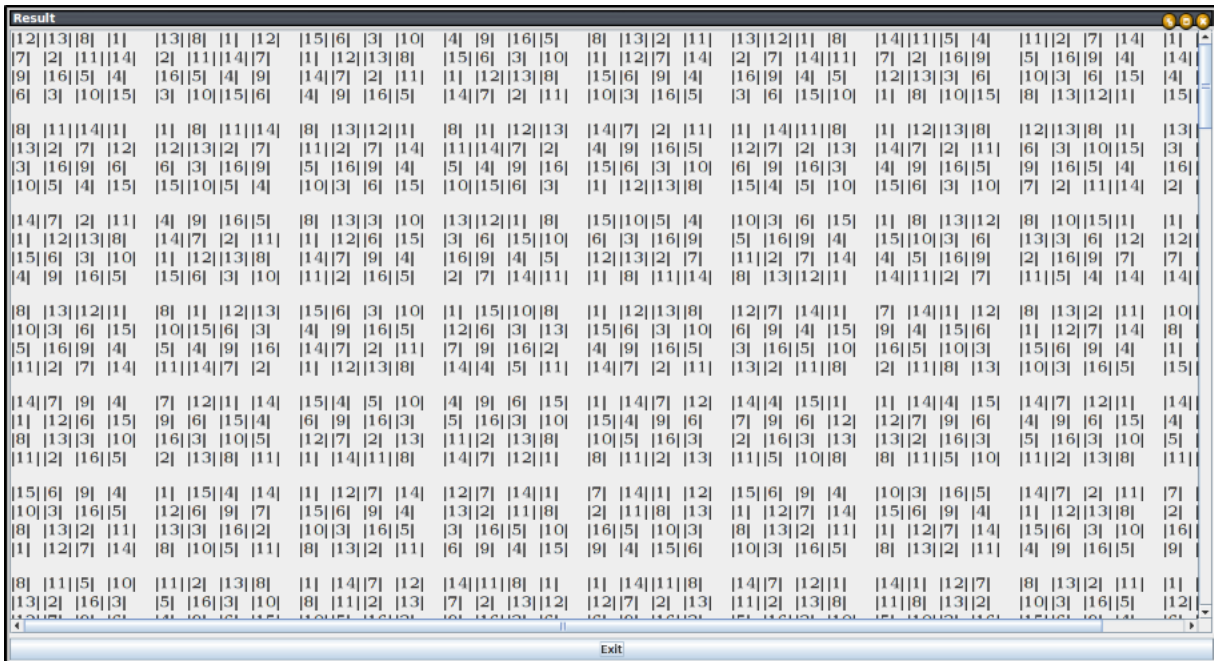
Figure 2: Principal Program Window .

- If you chose the option that generates squares, the program will be display a window with you request. See the Figure 3 and the Figure 4.



```
|1| |8| |10||15|   |1| |8| |11||14|   |1| |8| |13||12|   |1| |8| |10||15|
|12||13||3| |6|    |12||13||2| |7|    |14||11||2| |7|    |14||11||5| |4|
|7| |2| |16||9|    |6| |3| |16||9|    |4| |5| |16||9|    |7| |2| |16||9|
|14||11||5| |4|    |15||10||5| |4|    |15||10||3| |6|    |12||13||3| |6|
```

Figure 3: Window that show four squares.



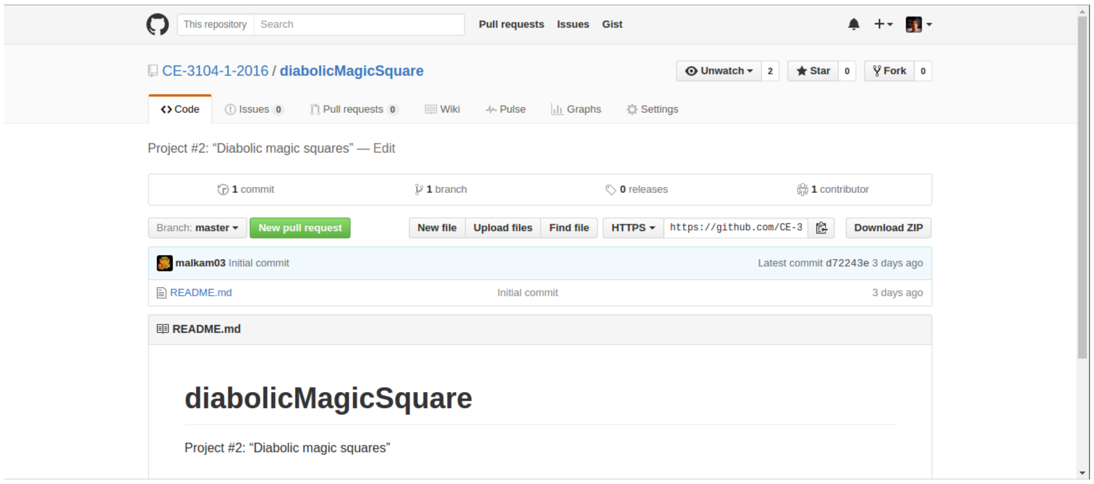
```
|12||13||8| |1|    |13||8| |1| |12|   |15||6| |3| |10|   |4| |9| |16||5|    |8| |13||2| |11|   |13||12||1| |8|    |14||11||5| |4|    |11||2| |7| |14|   |1|
|7| |2| |11||14|   |2| |11||14||7|    |1| |12||13||8|    |15||6| |3| |10|   |1| |12||7| |14|   |2| |7| |14||11|   |7| |2| |16||9|    |5| |16||9| |4|    |14|
|9| |16||5| |4|    |16||5| |4| |9|    |14||7| |2| |11|   |1| |12||13||8|    |15||6| |9| |4|    |16||9| |4| |5|    |12||13||3| |6|    |10||3| |6| |15|   |4|
|6| |3| |10||15|   |3| |10||15||6|    |4| |9| |16||5|    |14||7| |2| |11|   |10||3| |16||5|    |3| |6| |15||10|   |1| |8| |10||15|   |8| |13||12||1|    |15|

|8| |11||14||1|    |1| |8| |11||14|   |8| |13||12||1|    |8| |1| |12||13|   |14||7| |2| |11|   |1| |14||11||8|    |1| |12||13||8|    |12||13||8| |1|    |13|
|13||2| |7| |12|   |12||13||2| |7|    |11||2| |7| |14|   |11||14||7| |2|    |4| |9| |16||5|    |12||7| |2| |13|   |14||7| |2| |11|   |6| |3| |10||15|   |3|
|3| |16||9| |6|    |6| |3| |16||9|    |5| |16||9| |4|    |5| |4| |9| |16|   |15||6| |3| |10|   |6| |9| |16||3|    |4| |9| |16||5|    |9| |16||5| |4|    |16|
|10||5| |4| |15|   |15||10||5| |4|    |10||3| |6| |15|   |10||15||6| |3|    |1| |12||13||8|    |15||4| |5| |10|   |15||6| |3| |10|   |7| |2| |11||14|   |2|

|14||7| |2| |11|   |4| |9| |16||5|    |8| |13||3| |10|   |13||12||1| |8|    |15||10||5| |4|    |10||3| |6| |15|   |1| |8| |13||12|   |8| |10||15||1|    |1|
|1| |12||13||8|    |14||7| |2| |11|   |1| |12||6| |15|   |3| |6| |15||10|   |6| |3| |16||9|    |5| |16||9| |4|    |15||10||3| |6|    |13||3| |6| |12|   |12|
|15||6| |3| |10|   |1| |12||13||8|    |14||7| |9| |4|    |16||9| |4| |5|    |12||13||2| |7|    |11||2| |7| |14|   |4| |5| |16||9|    |2| |16||9| |7|    |7|
|4| |9| |16||5|    |15||6| |3| |10|   |11||2| |16||5|    |2| |7| |14||11|   |1| |8| |11||14|   |8| |13||12||1|    |14||11||2| |7|    |11||5| |4| |14|   |14|

|8| |13||12||1|    |8| |1| |12||13|   |15||6| |3| |10|   |1| |15||10||8|    |1| |12||13||8|    |12||7| |14||1|    |7| |14||1| |12|   |8| |13||2| |11|   |10|
|10||3| |6| |15|   |10||15||6| |3|    |4| |9| |16||5|    |12||6| |3| |13|   |15||6| |3| |10|   |6| |9| |4| |15|   |9| |4| |15||6|    |1| |12||7| |14|   |8|
|5| |16||9| |4|    |5| |4| |9| |16|   |14||7| |2| |11|   |7| |9| |16||2|    |4| |9| |16||5|    |3| |16||5| |10|   |16||5| |10||3|    |15||6| |9| |4|    |1|
|11||2| |7| |14|   |11||14||7| |2|    |1| |12||13||8|    |14||4| |5| |11|   |14||7| |2| |11|   |13||2| |11||8|    |2| |11||8| |13|   |10||3| |16||5|    |15|

|14||7| |9| |4|    |7| |12||1| |14|   |15||4| |5| |10|   |4| |9| |6| |15|   |1| |14||7| |12|   |14||4| |15||1|    |1| |14||4| |15|   |14||7| |12||1|    |14|
|1| |12||6| |15|   |9| |6| |15||4|    |6| |9| |16||3|    |5| |16||3| |10|   |15||4| |9| |6|    |7| |9| |6| |12|   |12||7| |9| |6|    |4| |9| |6| |15|   |4|
|8| |13||3| |10|   |16||3| |10||5|    |12||7| |2| |13|   |11||2| |13||8|    |10||5| |16||3|    |2| |16||3| |13|   |13||2| |16||3|    |5| |16||3| |10|   |5|
|11||2| |16||5|    |2| |13||8| |11|   |1| |14||11||8|    |14||7| |12||1|    |8| |11||2| |13|   |11||5| |10||8|    |8| |11||5| |10|   |11||2| |13||8|    |11|

|15||6| |9| |4|    |1| |15||4| |14|   |1| |12||7| |14|   |12||7| |14||1|    |7| |14||1| |12|   |15||6| |9| |4|    |10||3| |16||5|    |14||7| |2| |11|   |7|
|10||3| |16||5|    |12||6| |9| |7|    |15||6| |9| |4|    |13||2| |11||8|    |2| |11||8| |13|   |1| |12||7| |14|   |15||6| |9| |4|    |1| |12||13||8|    |2|
|8| |13||2| |11|   |13||3| |16||2|    |10||3| |16||5|    |3| |16||5| |10|   |16||5| |10||3|    |8| |13||2| |11|   |1| |12||7| |14|   |15||6| |3| |10|   |16|
|1| |12||7| |14|   |8| |10||5| |11|   |8| |13||2| |11|   |6| |9| |4| |15|   |9| |4| |15||6|    |10||3| |16||5|    |8| |13||2| |11|   |4| |9| |16||5|    |9|

|8| |11||5| |10|   |11||2| |13||8|    |1| |14||7| |12|   |14||11||8| |1|    |1| |14||11||8|    |14||7| |12||1|    |14||1| |12||7|    |8| |13||2| |11|   |1|
|13||2| |16||3|    |5| |16||3| |10|   |8| |11||2| |13|   |7| |2| |13||12|   |12||7| |2| |13|   |11||2| |13||8|    |11||8| |13||2|    |10||3| |16||5|    |12|
```

Figure 4: Window that show all the squares.

4 Development environment

4.1 Git and Github

To store the code in a practical way and have a versioning control Git is used for creating repositories and GitHub as service and cloud storage. This code manager (Git) let us to create branches for each member of the group and work separately, change the branch work, see and edit code of another branches, make commits for later if you want to go back in your code and when each part needed is done they can be merged into master branch to have a final version of the code. See Fig 5



CE-3104-1-2016 / diabolicMagicSquare

Project #2: "Diabolic magic squares" — Edit

1 commit | 1 branch | 0 releases | 1 contributor

https://github.com/CE-3

malkam03 Initial commit — Latest commit d72243e 3 days ago

README.md — Initial commit — 3 days ago

README.md

diabolicMagicSquare

Project #2: "Diabolic magic squares"

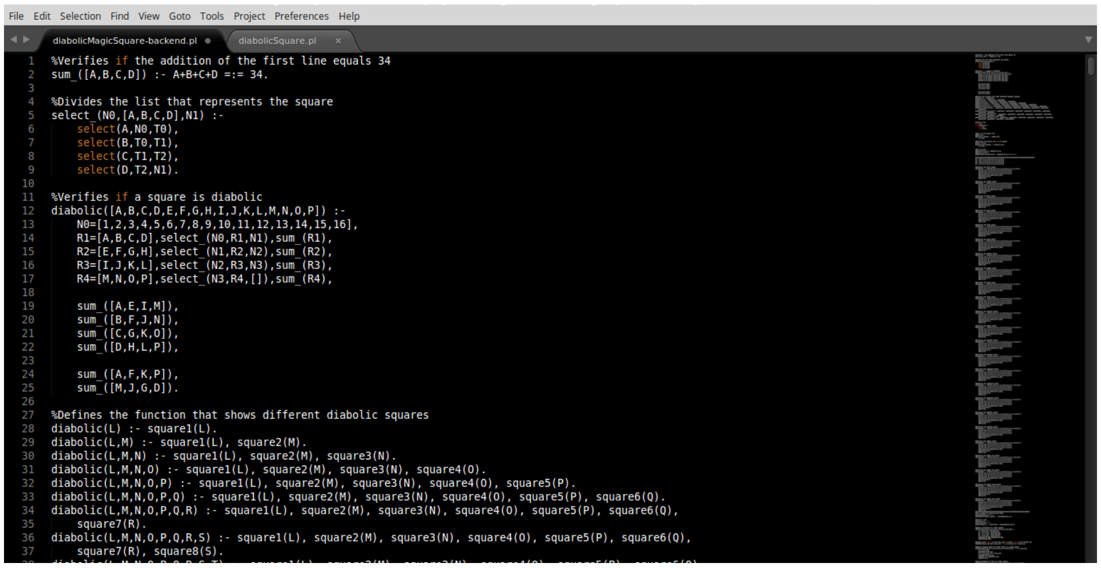
Figure 5: GitHub Interface.

4.2 SublimeText

It's a text editor for code, markup and prose. It contain functionalities very useful for programmers and facilitate the code development process. It proved to be very versatile and stable, giving good impressions, simple but with great features. Things like word suggestion, coloring restricted words, minimap, multi selection and multi cursor.Also support multiple languages include Prolog and others like Java, C, C++. See the Figure 6

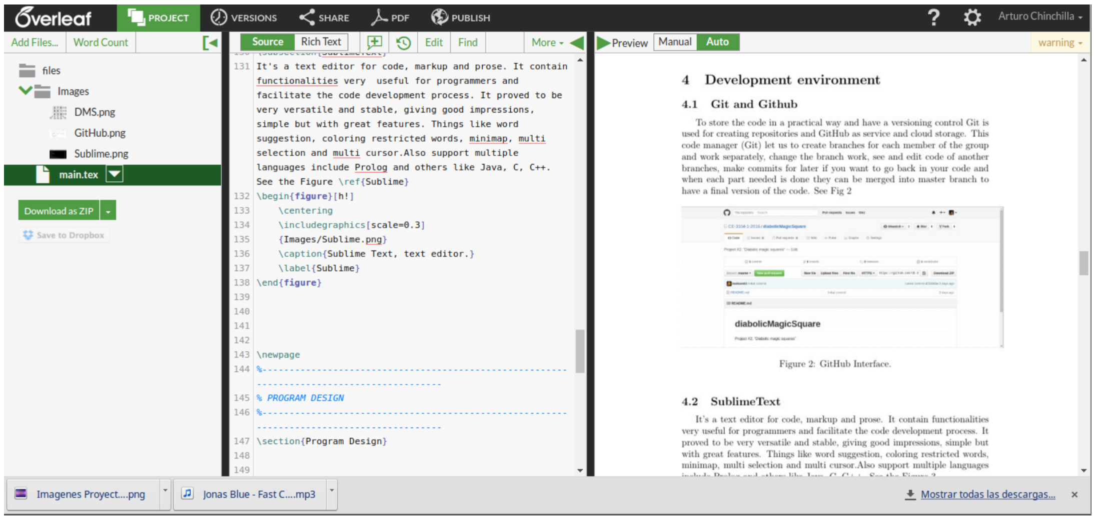
4.3 Overleaf

Free online tool used to write academic documents. This service allow create, edit and share your documents easily using LaTeX. Provide two principal windows, in the first you can write your document, obviously using the LaTeXsyntax, and the second you can see your compiled document. LaTeXalso is composed of a large set of macros getting documents have a typographic high quality. Therefore it is widely used for the creation of academic papers, theses and technical books, since the typographic quality and made with LaTeXdocuments is comparable to that of a great scientific publishing. See the Figure 7


File Edit Selection Find View Goto Tools Project Preferences Help
diabolicMagicSquare-backend.pl
diabolicSquare.pl

```
%Verifies if the addition of the first line equals 34
sum_([A,B,C,D]) :- A+B+C+D =:= 34.

%Divides the list that represents the square
select_(N0,[A,B,C,D],N1) :-
    select(A,N0,T0),
    select(B,T0,T1),
    select(C,T1,T2),
    select(D,T2,N1).

%Verifies if a square is diabolic
diabolic([A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P]) :-
    N0=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16],
    R1=[A,B,C,D],select_(N0,R1,N1),sum_(R1),
    R2=[E,F,G,H],select_(N1,R2,N2),sum_(R2),
    R3=[I,J,K,L],select_(N2,R3,N3),sum_(R3),
    R4=[M,N,O,P],select_(N3,R4,[]),sum_(R4),

    sum_([A,E,I,M]),
    sum_([B,F,J,N]),
    sum_([C,G,K,O]),
    sum_([D,H,L,P]),

    sum_([A,F,K,P]),
    sum_([M,J,G,D]).

%Defines the function that shows different diabolic squares
diabolic(L) :- square1(L).
diabolic(L,M) :- square1(L), square2(M).
diabolic(L,M,N) :- square1(L), square2(M), square3(N).
diabolic(L,M,N,O) :- square1(L), square2(M), square3(N), square4(O).
diabolic(L,M,N,O,P) :- square1(L), square2(M), square3(N), square4(O), square5(P).
diabolic(L,M,N,O,P,Q) :- square1(L), square2(M), square3(N), square4(O), square5(P), square6(Q).
diabolic(L,M,N,O,P,Q,R) :- square1(L), square2(M), square3(N), square4(O), square5(P), square6(Q),
    square7(R).
diabolic(L,M,N,O,P,Q,R,S) :- square1(L), square2(M), square3(N), square4(O), square5(P), square6(Q),
    square7(R), square8(S).
```

Figure 6: Sublime Text, text editor.



```
It's a text editor for code, markup and prose. It contain
functionalities very  useful for programmers and
facilitate the code development process. It proved to be
very versatile and stable, giving good impressions,
simple but with great features. Things like word
suggestion, coloring restricted words, minimap, multi
selection and multi cursor.Also support multiple
languages include Prolog and others like Java, C, C++.
See the Figure \ref{Sublime}
\begin{figure}[h!]
    \centering
    \includegraphics[scale=0.3]
    {Images/Sublime.png}
    \caption{Sublime Text, text editor.}
    \label{Sublime}
\end{figure}




\newpage
%-----------------------------------------------------------
---------------------------------
% PROGRAM DESIGN
%-----------------------------------------------------------
---------------------------------
\section{Program Design}
```

4 Development environment

4.1 Git and Github

To store the code in a practical way and have a versioning control Git is used for creating repositories and GitHub as service and cloud storage. This code manager (Git) let us to create branches for each member of the group and work separately, change the branch work, see and edit code of another branches, make commits for later if you want to go back in your code and when each part needed is done they can be merged into master branch to have a final version of the code. See Fig 2

diabolicMagicSquare

Figure 2: GitHub Interface.

4.2 SublimeText

It's a text editor for code, markup and prose. It contain functionalities very useful for programmers and facilitate the code development process. It proved to be very versatile and stable, giving good impressions, simple but with great features. Things like word suggestion, coloring restricted words, minimap, multi selection and multi cursor.Also support multiple languages

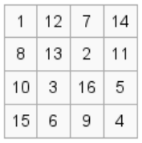
Figure 7: Overleaf Interface Code-Compiled.

5 Program Design

The Backend contains the logical part, it's in charge to check if the user entries is a Magic Square, and generates 384 different Magic Squares based in some Squares stored in the code. But, how is it possible? the squares are generated applying some operations to the matrices that represent the principal squares.

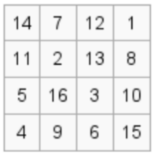
5.1 Reflection

The reflection operation consist in change the first column by the last column, generating a new magic square. See the Figure 8 and the Figure 9



1	12	7	14
8	13	2	11
10	3	16	5
15	6	9	4

Figure 8: Magic Square A.

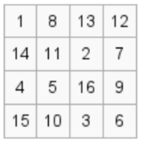


14	7	12	1
11	2	13	8
5	16	3	10
4	9	6	15

Figure 9: Reflection of A.

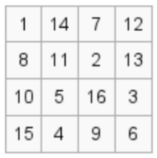
5.2 Rotation about the center point

This operation take the external values around the corners and change this. See the Figure 10 and the Figure 11



1	8	13	12
14	11	2	7
4	5	16	9
15	10	3	6

Figure 10: Magic Square B.



1	14	7	12
8	11	2	13
10	5	16	3
15	4	9	6

Figure 11: Rotation about the center of B.

5.3 Rotation of Columns

This operation get and remove the last column (most right column) and put it on the first place in the square.

5.4 Rotation of Rows

This operation get and remove the top row of the square and put it on the bottom place.

5.5 Convolution

This operation consist in divide the square in four mini-squares, and rotate those squares. See the Figure 12 and the Figure 13

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Figure 12: Magic Square C.

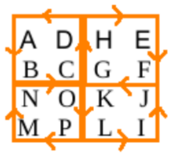
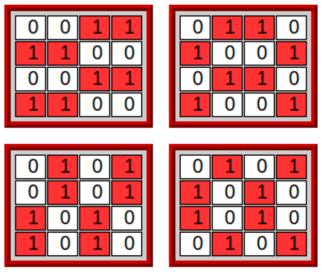


Figure 13: Convolution of C.

5.6 Solution

”All of the Order 4 Pan-Magic Squares are based on the same underlying ”Magic Carpet”, a simple pattern consisting of alternating pairs of ones with zeros(See the figure 14). Four samples of this pattern are multiplied by 8, 4, 2, and 1, to make the Magic Carpets which are added together to make the final square. The 8, 4, 2, and 1, can be used in any order to make different squares” [4].



0	0	1	1
1	1	0	0
0	0	1	1
1	1	0	0

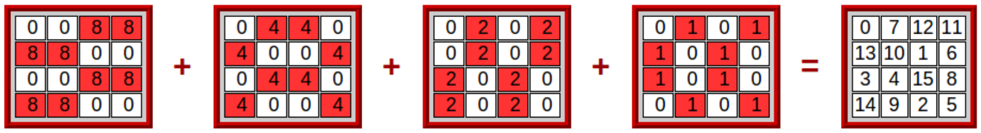
0	1	1	0
1	0	0	1
0	1	1	0
1	0	0	1

0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

0	1	0	1
1	0	1	0
1	0	1	0
0	1	0	1

Figure 14: Patterns of zeros and ones.

This sequence of squares show the production of one square by this technique: Figure 15



0	0	8	8
8	8	0	0
0	0	8	8
8	8	0	0

+

0	4	4	0
4	0	0	4
0	4	4	0
4	0	0	4

+

0	2	0	2
0	2	0	2
2	0	2	0
2	0	2	0

+

0	1	0	1
1	0	1	0
1	0	1	0
0	1	0	1

=

0	7	12	11
13	10	1	6
3	4	15	8
14	9	2	5

Figure 15: Patterns of zeros and ones.

Note: For our project, sum 1 to each value of the squares, because the range of possible values is 1 to 16.

6 Student Activity Log

6.1 Arturo’s Time-Sheet

Date	Task Description	Spent Time
04/12/2016	Meeting to divide tasks.	2 hours
04/12/2016	Installing SWI-Prolog.	2 hours
04/12/2016	Investigation about Prolog syntax	5 hours
04/13/2016	Create the verifies entry and auxiliaries	7 hours
04/13/2016	Implement the method that creates the principal squares	8 hours
04/14/2016	Implement the operations methods	10 hours
04/14/2016	Fixing some bugs	6 hours
04/15/2016	Create the method showall	13 hours
04/15/2016	Documentation starts	7 hours
	Total	60 hours

6.2 Malcolm's Time-Sheet

Date	Task Description	Spent Time
04/12/2016	Meeting to divide tasks	2 hours
04/12/2016	Installing SWI-Prolog.	2 hours
04/13/2016	Investigation about very easy language to connect both	4 hours
04/13/2016	Interface beginning using QT	7 hours
04/14/2016	Some more investigation about the language	3 hours
04/14/2016	Interface beginning again using Java	8 hours
04/14/2016	Implement the communication	6 hours
04/14/2016	Trying to fix the communication with the backend	5 hours
04/14/2016	Trying to fix the communication with the backend	7 hours
04/15/2016	Make the last fixes to the interface	9 hours
04/15/2016	Documentation starts	6 hours
	Total	61 hours

7 Project Final Status.

The project is completed in a good way, all the functionalities and interface developed successfully, and communication between the two sides also occurs successfully. The interface is friendly with the user and anyone can use it.

7.1 Issues, Limitations or Challenges

- From previous courses we bring the habit of thinking of a same way, but nevertheless in the course of Programming Languages we are forced to learn new paradigms, and thus think of a different way to solve computational problems , the difficulty of learning to think in a differently way (new programming paradigm), in addition to learning a new language with its syntax, semantics and pragmatics.
- Choose the correct Programming language that functions in conjunction with Prolog was a very hard decision, all the investigation and the time of testing and error.
- Connect the Backend with the Frontend was difficult, because both was developed on different programming languages, and for this many times the languages behavior wasn't correct..

7.2 known Issues

- When one square is requested for the user, the program takes some time to respond to the request,for that reason the option is disabled on the interface.

8 Conclusions, Suggestions and Recommendations.

- Tools like the handler code versions is very useful, because in case of any failure in the new version allows us to return to a previous, plus the possibility of linking it to the storage service in the cloud, to share, and work together in a very easy way.
- Sublime Text can be one of the better text editors, that should not make lack in the programmer's computer, for its flexibility and support for multiple programming languages, in addition to the vast amount of functionalities adapted for this use.
- Connect the Frontend with the Backend wasn't easy, because use two different language make it difficult, the only way is use external libraries that help with this task.

References

[1] R. Barták, *"Prolog Guide - Introduction"* Kti.ms.mff.cuni.cz, 1998. [Online]. Available: http://kti.ms.mff.cuni.cz/ bartak/prolog/intro.html. [Accessed: 13- Apr- 2016].

[2] W. Andrews, *"Magic squares and cubes"* New York: Dover Publications, 1960, p. 1.

[3] *"Math Logic - Diabolic Magic Square"* Calculatoredge.com, 2008. [Online]. Available: http://www.calculatoredge.com/math/mathlogic/logicans22.htm. [Accessed: 16- Apr- 2016].

[4] *"The 4x4 Pan-Magic Squares"* Grogono.com, 2010. [Online]. Available: http://www.grogono.com/magic/4x4.php. [Accessed: 16- Apr- 2016].

[5] *"Past programming languages and their influences on today's languages and programming paradigms - Engineering and Technology History Wiki"* Ethw.org, 2015. [Online]. Available: http://ethw.org/Past_programming_languages_and_their_influences_on_today's_languages_and_ [Accessed: 16- Apr- 2016].