Rajiv Gandhi University of Knowledge Technologies

R.K Valley, Y.S.R Kadapa (Dist)-516330

A
project report
on

Sudoku Solver

Submitted by

T.HEMA SREE R170549
P.NISHA BANU R171211
M.ASHA PRAVEENA R171203

Under the guidance of

K.VINOD KUMAR SIR

Department of Computer Science Engineering

This project report has been submitted in fulfillment of the requirements for the Degree of Bachelor of Technology in software Engineering.

March-2023

Rajiv Gandhi University of Knowledge Technologies
IIIT,R.K.Valley,YSR Kadapa(Dist)-516330

CERTIFICATE

This is to certify that report entitled "**Sudoku Solver"** Submitted by T.Hemasree(R170549),M>Asha Pravenna(R171203) , P.Nish Bhanu(R171211) in partial fulfillment of the requirements of the award of bachelor of technology in Computer Science Engineering is a bonafide work carried by the them under the supervision and guidance.

The report has been not submitted previously in part or full to this or any other university or institue for the award of any degree or diploma.

GUIDE	HEAD OF THE DEPARTMENT
K.Vinod Kumar	Mr.N.Satyanandram

ACKNOWLEDGEMENT

The satisfaction that accompanies the succeful completion of any task would be incomplete without the mention of the people who made it possible and who's constant guidance and encouragement crown all the efforts success.

I would like to express my sincere gratitude to **K.VINOD KUMAR sir** , my project guide for valuable suggestions and keen interest throughout the progress of my project.

I am grateful to **Mr.N.Satyanandram sir HOD CSE** ,for providing excellent computing facilities and congenial atmosphere for progressing my project.

At the outset,I would like to thank **Rajiv Gandhi Of University of Knowledge Technologies(RGUKT),**for providing all the necessary resources and support for the successful completion of my course work.

DECLARATION

We hereby declare that this report entitled“**Sudoku Solver ”** Submitted by us under the guidance and Supervision o**f K.Vinod KUMAR** sir ,is a bonafide work.we also declare that it has not been of Submitted previously in part or in full to this University or other institution for the award of any degree or diploma.

Date:- 28-04-2023 T>Hema sree(R170549)

Place:-RK VALLEY M.AshaPraveena(R171203)

P.Nisha Bhanul(R171211)

INDEX

ABSTRACT

In the last decade, solving the Sudoku puzzle has become every one’s passion. The simplicity of puzzle’s structure and the low requirement of mathematical skills caused people to have enormous interest in accepting challenges to solve the puzzle. Therefore, developers have tried to find algorithms in order to generate the variety of puzzles for human players so that they could be even solved by computer programming. In this essay, we have presented an algorithm called pencil-and-paper using human strategies. The purpose is to implement a more efficient algorithm and then compare it with another Sudoku solver named as brute force algorithm. This algorithm is a general algorithm that can be employed in to any problems.

1 Introduction

Currently, Sudoku puzzles are becoming increasingly popular among the people all over the world. The game has become popular now in a large number of countries and many developers have tried to generate even more complicated and more interesting puzzles. Today, the game appears in almost every newspaper, in books and in many websites.

In this essay we present a Sudoku Solver named as pencil-and-paper algorithm using simple rules to solve the puzzles. The pencil-and-paper algorithm is formulated based on human techniques. This means that the algorithm is implemented based on human perceptions. Therefore the name of the solver is pencil-and-paper algorithm. The Brute force algorithm is then used to compare with this algorithm in order to evaluate the efficiency of the proposed algorithm. The brute force is a general algorithm than can be applied to any possible problem. This algorithm generates any possible solutions until the right answer is found.
There are currently different variants of Sudoku such as 4X4 grids, 9X9 grids and 16X16 grids. This work is focused on classic and regular Sudoku of 9X9 board, and then a comparison is performed between the paper-and-pencil method and Brute force algorithm.

SOFTWARE REQUIREMENTS

Operating System:-Windows/Linux

Random Access Memory:-Min 4 GB

Technologies Used:Html,CSS,JavaScript.

HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.[3] This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

JavaScript

JavaScript often abbreviated to JS, is a programming language that is one of the core technologies of the World Wide Web, alongsideHTML and CSS.As of 2022, 98% of websites use JavaScript on the client sidefor webpage behavior,often incorporating third-party libraries.All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.Image result for wikipedia javascript
JavaScript is also used outside of web browsers. As a scripting language, JavaScript can be used to define the behaviour of applications such as extensions in GNOME Shell. In addition, there are runtime environments for running JavaScript as a server side programming language. Such an environment is Node.

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
1			9				7	2	8
2	2	7	8			3		1	
3		9					6	4	
4		5			6		2		
5			6				3		
6		1			5				
7	1			7		6		3	4
8				5		4			
9	7		9	1			8		5

Fig.1. *An example of a Sudoku puzzle.*

Purpose

The aim of the essay is to investigate the brute force algorithm and the pencil-and-paper algorithm. Later these two methods are compared analytically. It is expected here to find an efficient method to solve the Sudoku puzzles . In this essay we have tried to implement the pencil-and-paper algorithm that simulate how human being would solve the puzzle by using some simple strategies that can be employed to solve the majority of Sudoku.

Abbreviations and Definitions

In this essay we have tried to use the same terminology, which is commonly used in other journals and research papers. In the following paragraph, there is a brief description of some the abbreviations and definitions that are used in the text.

Sudoku: is a logic-based, combinatorial number placement puzzle [4]. The word "Sudoku" is short for Su-ji wa dokushin ni kagiru (in Japanese), which means "the numbers must be single", see Fig.1.

Box (Region, Block)**:** a region is a 3x3 box like the one shown in figure 1. There are 9 regions in a traditional Sudoku puzzle.

Cell (Square)**:** is used to define the minimum unit of the Sudoku board.

Candidates: the number of possible values that can be placed into an empty square.

Grid (board)**:** the Sudoku board consists of a form of matrix or windows.

is when the row, column and box hold 8 different numbers and one single number is left for that square, see Fig. 2.

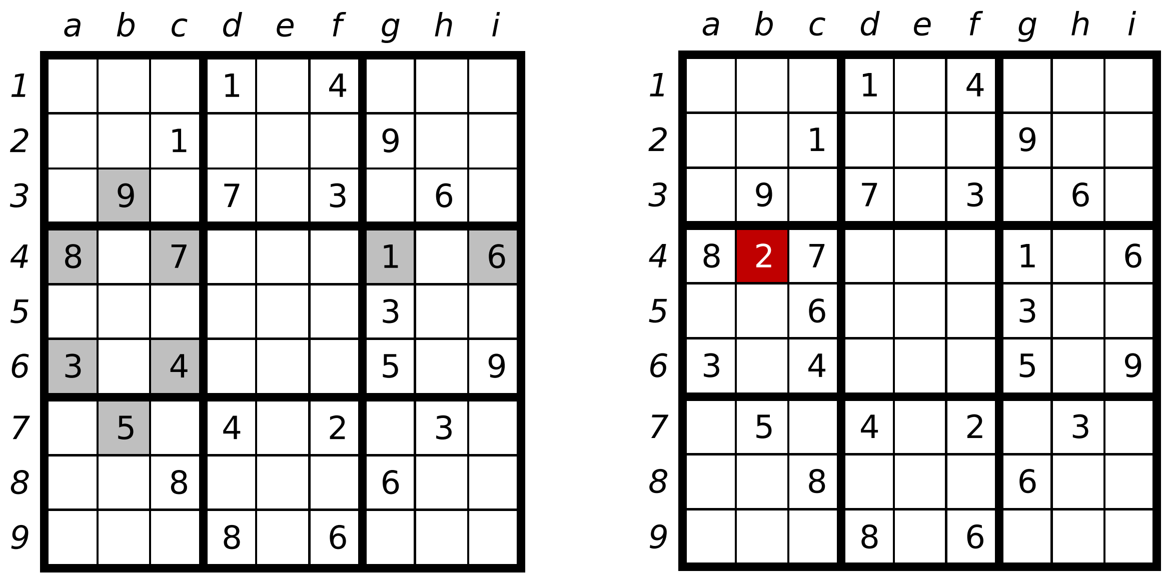


Fig. 2. *A description of the naked single method. In the left figure square 4b can hold just one possible number, which is 2 as it is inserted in the right figure.*

As we see in figure 2, it is possible to list all the candidates from 1 to 9 in each unfilled square, i.e. square 4b can only hold number 2 since it is the only candidate for this position. The most significant aspect is that when a candidate is found for a certain position then it can be removed from the list as a possible candidate in the row, column and box [7]. The reason that it is called the "naked single" method is that this kind of square contains only one possible candidate.

Hidden Singles

The hidden single method is similar to the naked single method but the way to find the way to find the empty square is different. When there is only one square in the row, column or box that can take a certain number, then the square must take that number. For example in figure 3, we can see that both row2 and row3 contain the digit 9 so according to the rules, row1 must also hold number 9 (in the square 123def). In the right side of figure 3 below, number 9 is inserted by using the hidden singles method.

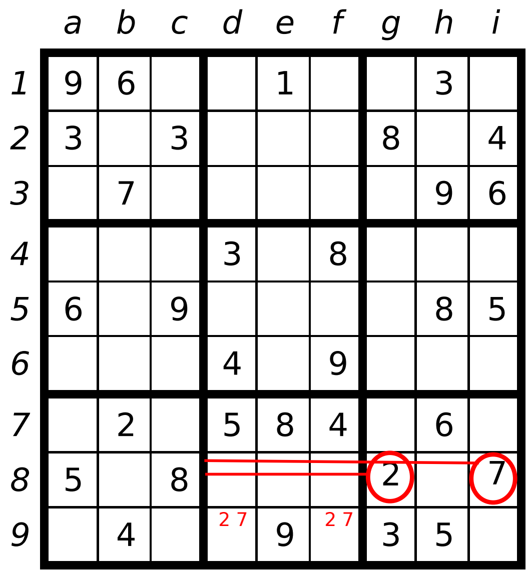


Fig. 4. *An illustration of naked pair technique [8].*

Brute force algorithm

Kovacs describe some of the brute force methods used for solving Sudoku puzzles [9]. The simplest method randomly produces a solution to the puzzle called “unconstrained grid”, after that the program checks whether it is a valid solution. If not, the process is repeated until a solution is found. This algorithm can be applied simply and will find a valid solution for any problems because it will go through all possibility solutions. However, this method can be time consuming but according to Kovacs the algorithm can be optimized.

Generally, the brute force algorithm goes through the empty squares, filling in numbers from the existing choices, or removing failed choices if a “dead-end” is reached. For example, Brute force solve a puzzle by inserting the digit “1” in the first square. If the digit is allowed to be there by checking row, column and box then the program go to the next square, and put the digit “1” in that square. The program discovers that the “1” is not allowed, then the digit increments by one i.e. it has become 2. When a square is noticed where none of the digits (1 to 9) is permitted, then the program backtracks and comes back to the prior square. The value in that square increases by 1. The process is repeated until the correct digits fill all 81 squares.

Approach

Here we carried out a study on how to solve Sudoku puzzle based on the pencil-and-paper algorithm. The concept of this work is to implement a solution to solve the puzzle based on human strategies. The obtained results are compared with a well-known algorithm called Brute force algorithm that are presented in chapter 3. The final result and conclusions are presented in chapter 4.

3 Analysis and Results

This section starts with analysis and discussions about two mentioned algorithms. A comparison is carried out between two algorithms in order to find out which algorithm is more efficient. At the end of the section, there are discussions on difficulty level of the puzzles and time complexity.

Unique missing candidate

This method is useful when there is just only one empty square in a row, column or box. The digit that is missing can be placed in that empty square. A similar definition is that if eight of nine empty squares are filled in any row, column or box, then the digit that is missing can fill the only empty square. This method can be useful when most of the squares are filled, especially at the end of a solution. It can also be suitable when solving easy puzzle and this method is efficient to find solution in this case. In this algorithm, the method goes through all rows, columns and boxes separately. The method then checks if a single value has missed in any row, column or box and place the single digit in that specific square (see the appendix).

Backtracking (guessing method)

The unique missing method and the naked single method are able to solve all puzzles with easy and medium level of difficulties. In order to solve puzzles with even more difficult levels such as hard and evil the backtracking method has been used to complete the algorithm. A human player solves the puzzle by using simple techniques. If the puzzle is not solvable by using the techniques the player then tries to fill the rest of the empty squares by guessing.

The backtracking method, which is similar to the human strategy (guessing), is used as a help method to the pencil-and-paper algorithm. In other words, if the puzzle cannot be filled when using the unique missing method and the naked single method, the backtracking method will take the puzzle and fill the rest of empty squares. Generally, the backtracking method find empty square and assign the lowest valid number in the square once the content of other squares in the same row, column and box are considered. However, if none of the numbers from 1 to 9 are valid in a certain square, the algorithm backtracks to the previous square, which was filled recently.

```
            recursiveBacktrackning(Puzzle[][]){

    Puzzle [][]//global

          solvePuzzle(row,col){
                      if (no more choices): the puzzle is solved!
                      If (puzzle[row][col]= notEmpty):
                             move to the next square.
for 1 to 9: if(checkRow(row,col,digit) & checkCol(row,col,digit) & checkBox(row,col,digit){
puzzle[row][col]= digit;
                             move to the next square
                             }
                    if not valid number is found go the previous square that was recently filled
                    }
```

Brute Force Solver

The second algorithm that is examined in this work is Brute force algorithm. Usually, the brute force algorithm can be applied to any possible algorithm. For example when finding password, the algorithm generates any possible password until the right one is found. In this case the algorithm goes through every empty square and places a valid digit in that square. If no valid number is found the algorithm comes back to the previous square and change the value in that square. The process is repeated until the board is filled with numbers from 1 to 9.

The advantage of the brute force algorithm is that the algorithm can guarantee a solution to any puzzles since it generates all possible answers until the right answer is found if the puzzles are valid [9]. Additionally, the running time can be unrelated to level of difficulty, because the algorithm searches for every possible solution.

In order to compare the pencil-and-paper algorithm with the brute force algorithm a (pre-implemented) brute force algorithm has been used during testing [10].

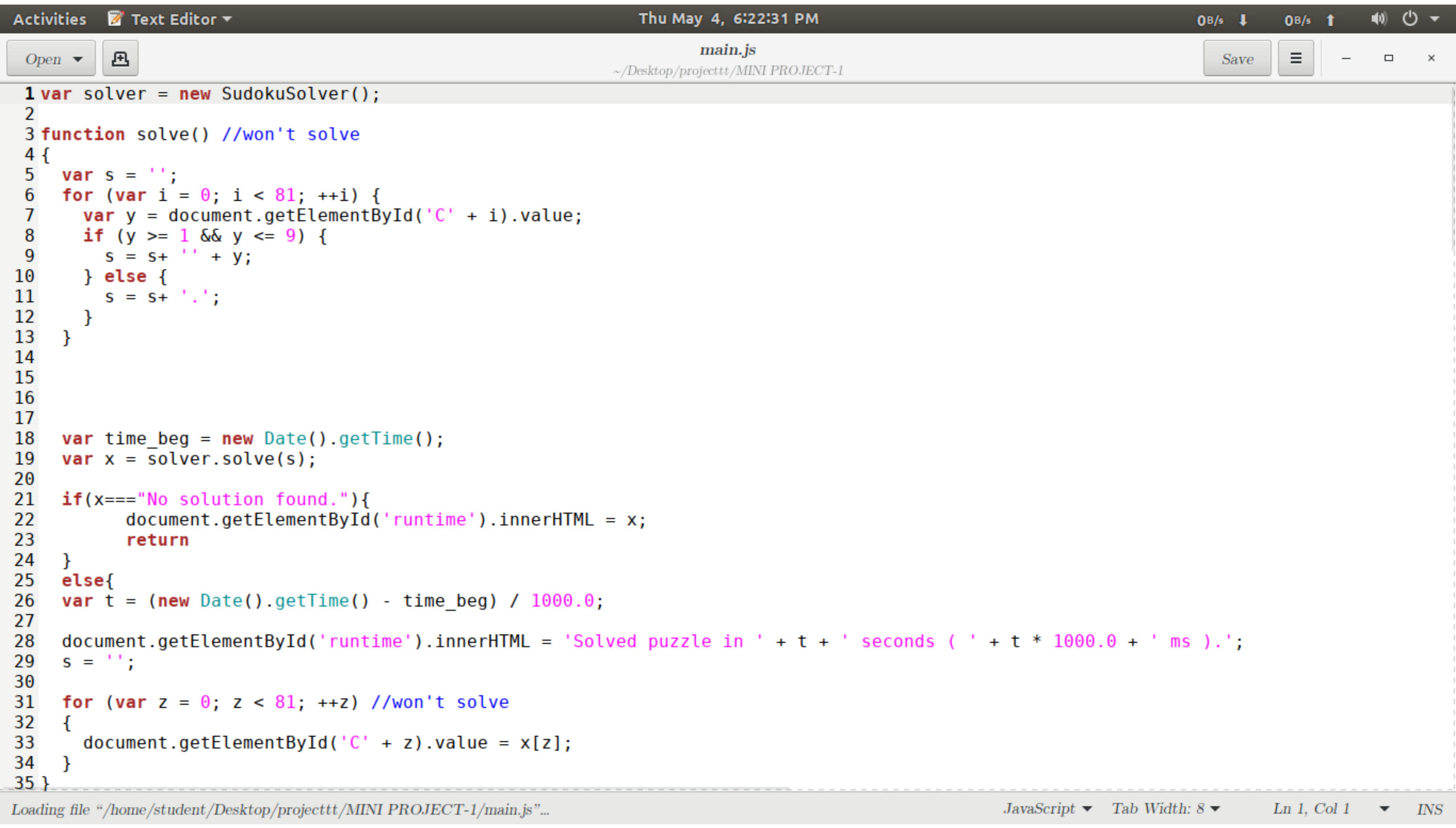
The Difficulty Level of Sudoku Puzzles

The difficulty level of Sudoku puzzles depends on how the given numbers are placed in the Sudoku board and also how many numbers (clues) are given. Generally, the most significant aspect of difficulty ratings of Sudoku puzzles is that which techniques are required to solve The Puzzles, which needs more techniques to solve, can be named as difficult one. On the other side there are puzzles that can be solved by using simple methods and this kind of puzzles can be defined as easy or medium level. As mentioned above, there are four difficulty levels that are used in testing (easy, medium, hard). We have found during testing that the classifications of difficulty levels is not easy as it is stated. This is due to the fact that there have been puzzles which marked as hard level but they had been solved using simple techniques and vice versa.

There are people who believe that the difficulty ratings have to do with the number of revealed numbers on the puzzle board. Generally, a Sudoku puzzle needs at least 17 clues to be solvable. It means that solving a Sudoku puzzle with 17-clues is more difficult than a puzzle with 30-clues. More given numbers, the easier and quicker the solution is. This statement may not be always truth, since the testing has shown that the puzzles with fewer clues could be solved in shorter time than the puzzles with more clues. The diagram 3 below describes the relation between the number of clues in the puzzles and their run-time. It is the puzzles. In other words, it is important where the given numbers are placed logically.

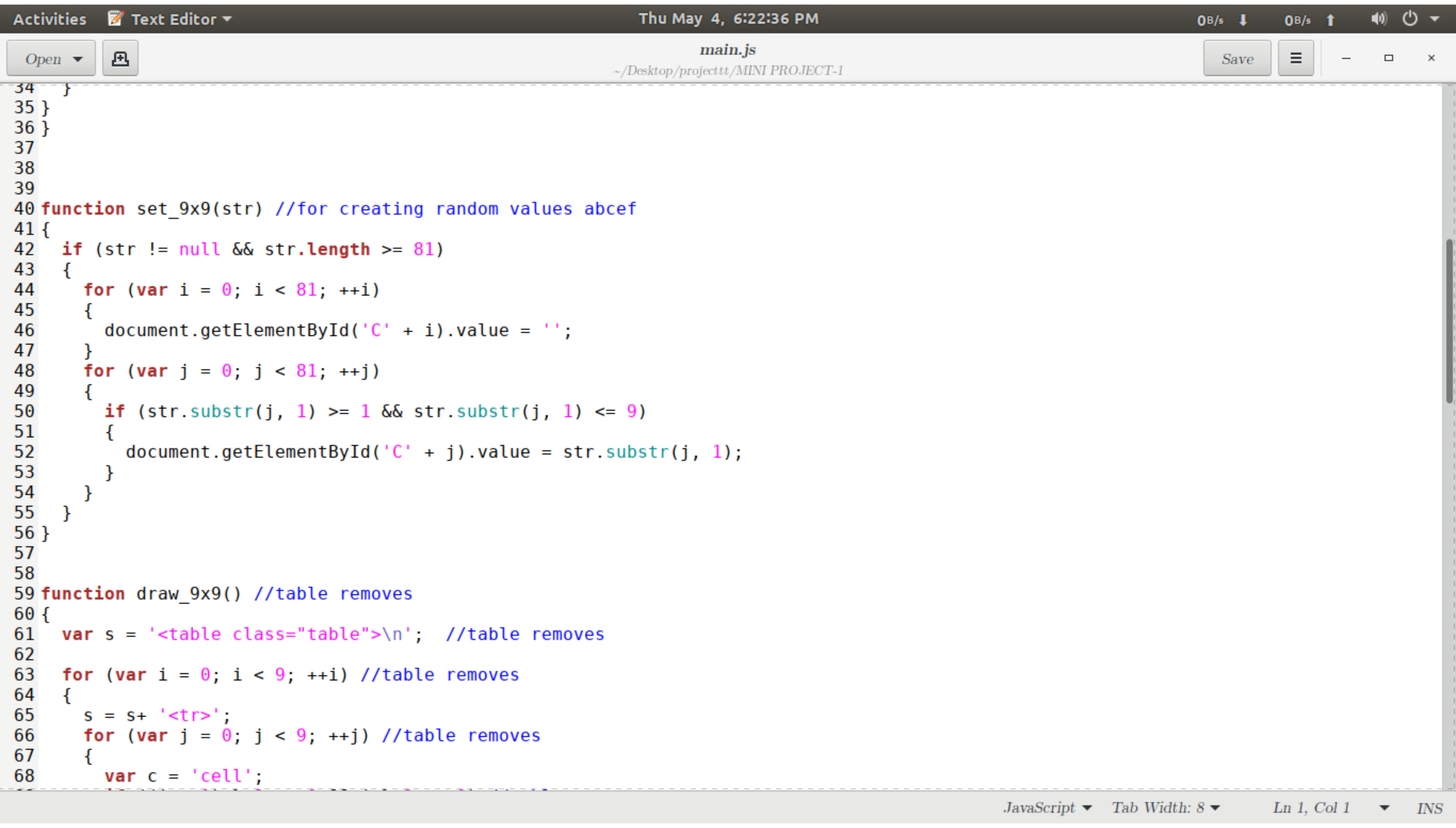
SOURCE CODE :

main.html

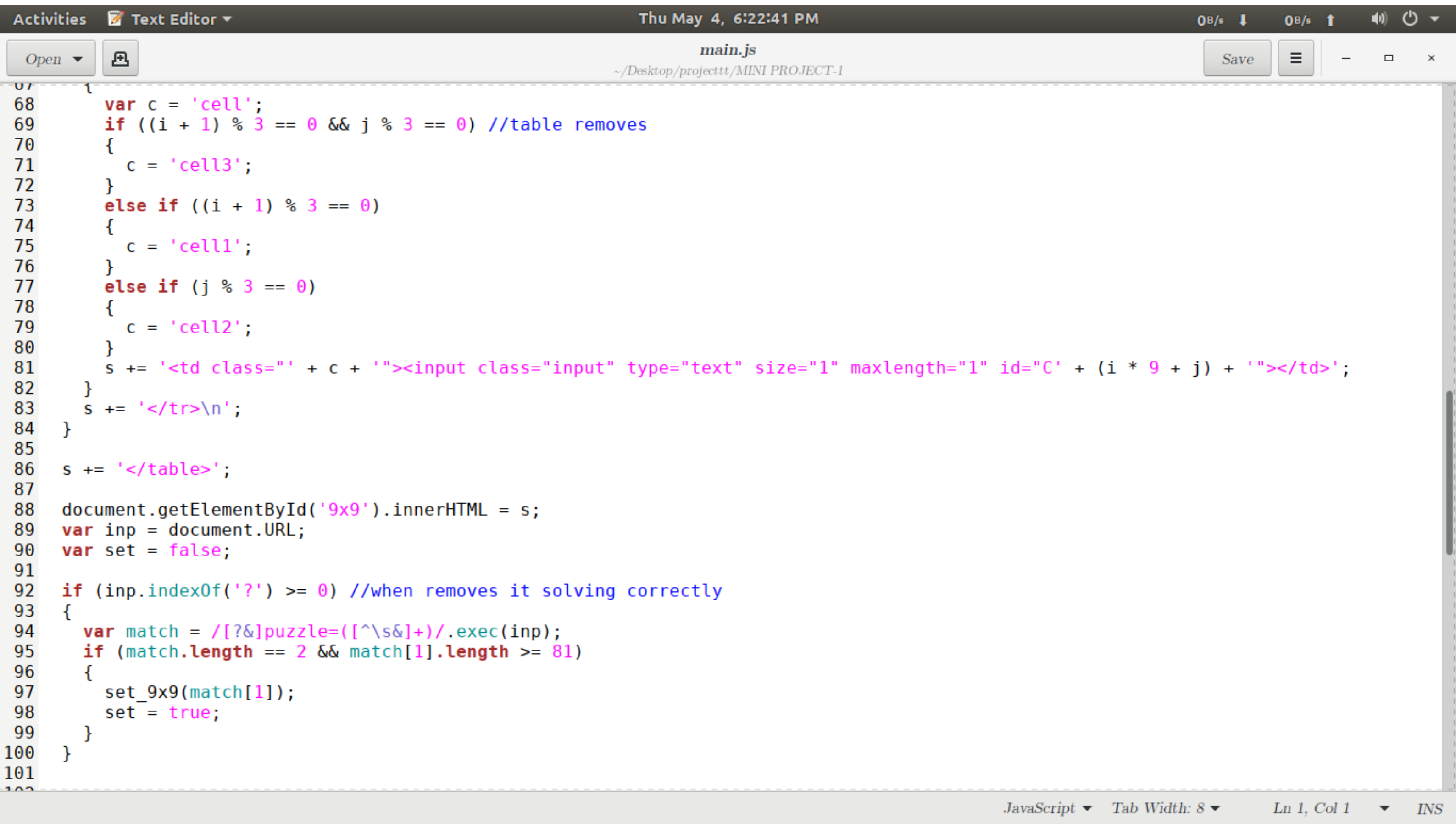


```
var solver = new SudokuSolver();

function solve() //won't solve
{
  var s = '';
  for (var i = 0; i < 81; ++i) {
    var y = document.getElementById('C' + i).value;
    if (y >= 1 && y <= 9) {
      s = s+ '' + y;
    } else {
      s = s+ '.';
    }
  }




  var time_beg = new Date().getTime();
  var x = solver.solve(s);

  if(x==="No solution found."){
        document.getElementById('runtime').innerHTML = x;
        return
  }
  else{
  var t = (new Date().getTime() - time_beg) / 1000.0;

  document.getElementById('runtime').innerHTML = 'Solved puzzle in ' + t + ' seconds ( ' + t * 1000.0 + ' ms ).';
  s = '';

  for (var z = 0; z < 81; ++z) //won't solve
  {
    document.getElementById('C' + z).value = x[z];
  }
}
```

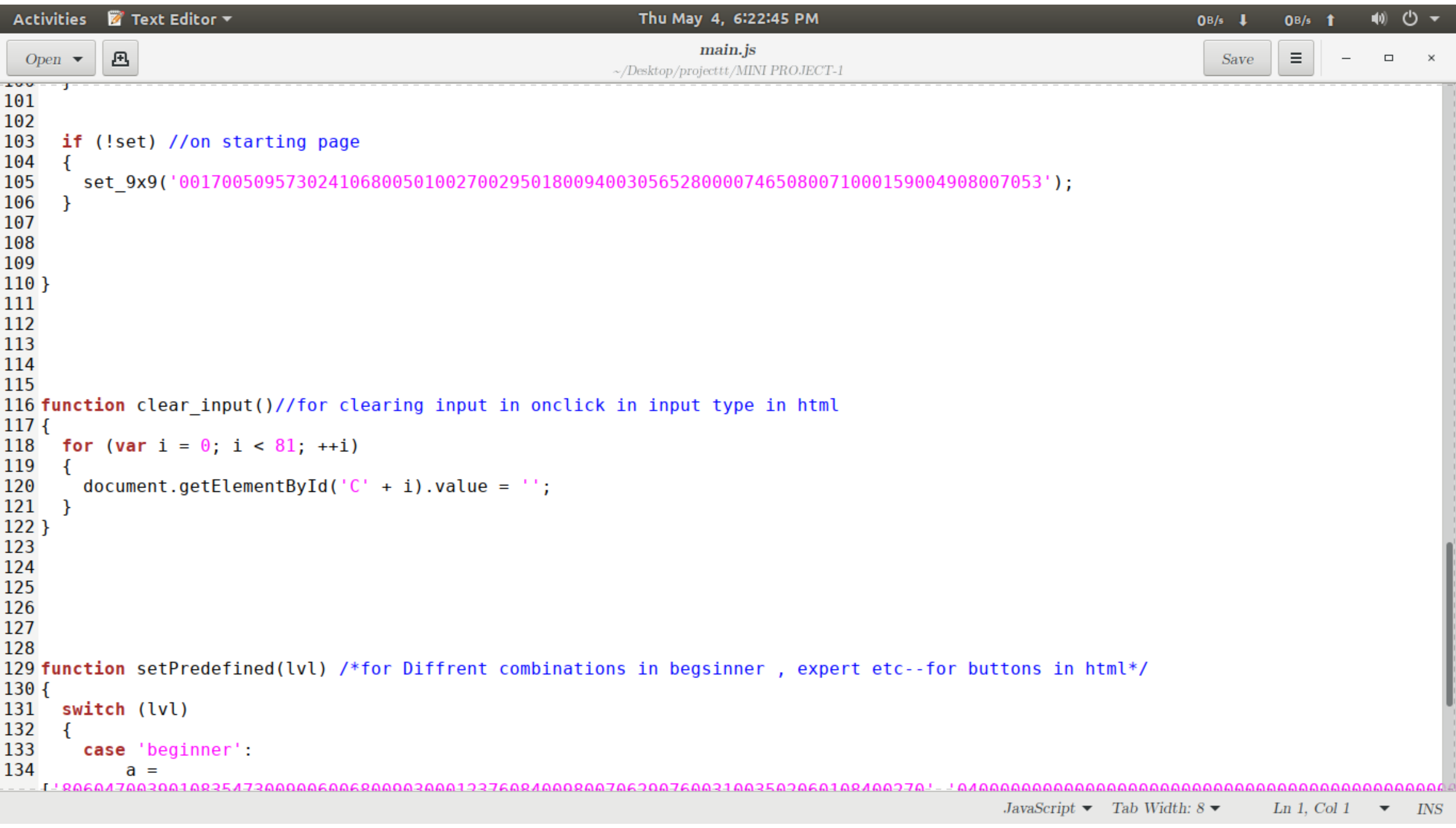
Loading file "/home/student/Desktop/projecttt/MINI PROJECT-1/main.js"...



```
  }
}
}



function set_9x9(str) //for creating random values abcef
{
  if (str != null && str.length >= 81)
  {
    for (var i = 0; i < 81; ++i)
    {
      document.getElementById('C' + i).value = '';
    }
    for (var j = 0; j < 81; ++j)
    {
      if (str.substr(j, 1) >= 1 && str.substr(j, 1) <= 9)
      {
        document.getElementById('C' + j).value = str.substr(j, 1);
      }
    }
  }
}


function draw_9x9() //table removes
{
  var s = '<table class="table">\n';  //table removes

  for (var i = 0; i < 9; ++i) //table removes
  {
    s = s+ '<tr>';
    for (var j = 0; j < 9; ++j) //table removes
    {
      var c = 'cell';
```

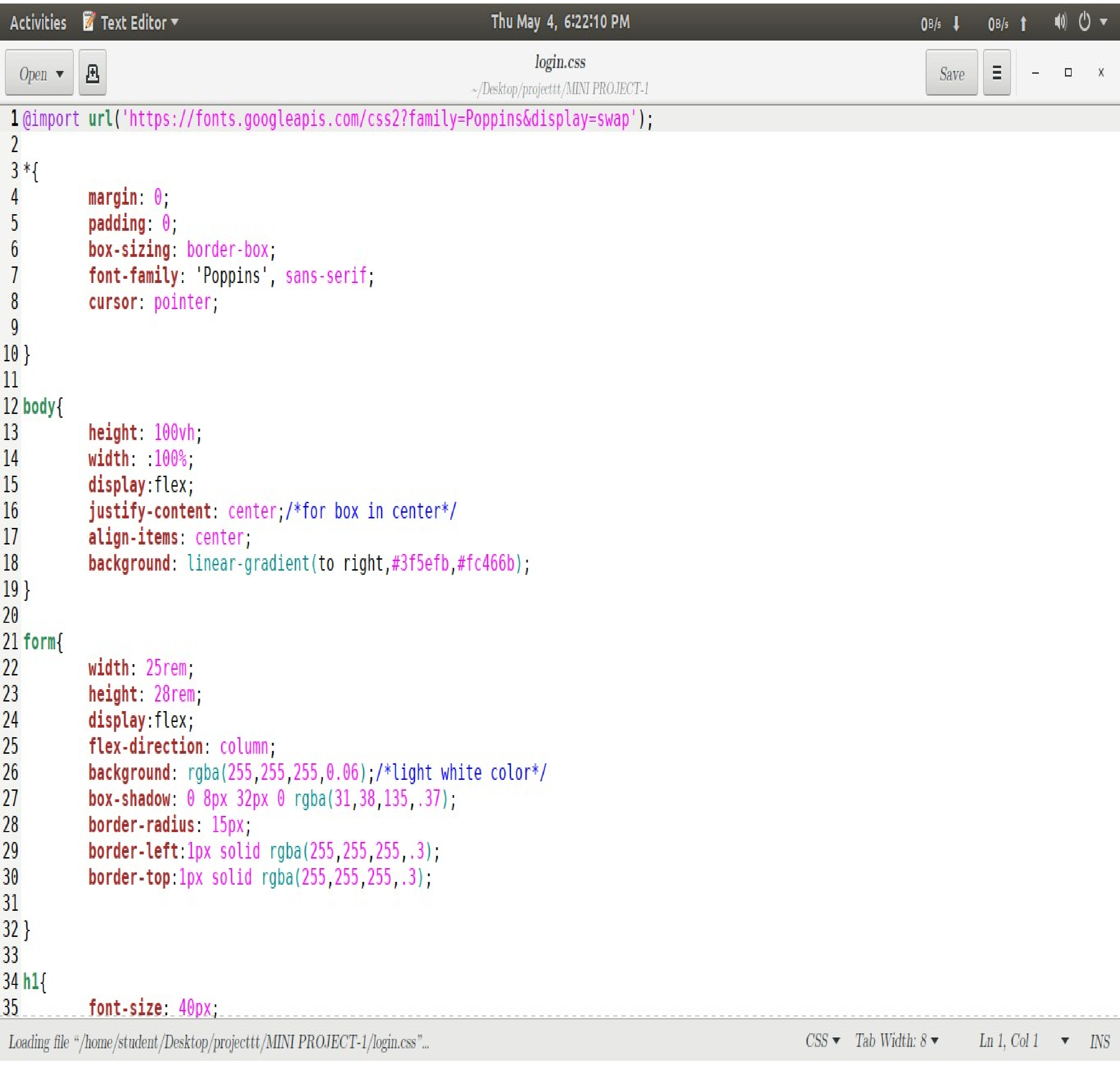


```
      var c = 'cell';
      if ((i + 1) % 3 == 0 && j % 3 == 0) //table removes
      {
        c = 'cell3';
      }
      else if ((i + 1) % 3 == 0)
      {
        c = 'cell1';
      }
      else if (j % 3 == 0)
      {
        c = 'cell2';
      }
      s += '<td class="' + c + '"><input class="input" type="text" size="1" maxlength="1" id="C' + (i * 9 + j) + '"></td>';
    }
    s += '</tr>\n';
  }

  s += '</table>';

  document.getElementById('9x9').innerHTML = s;
  var inp = document.URL;
  var set = false;

  if (inp.indexOf('?') >= 0) //when removes it solving correctly
  {
    var match = /[?&]puzzle=([^\s&]+)/.exec(inp);
    if (match.length == 2 && match[1].length >= 81)
    {
      set_9x9(match[1]);
      set = true;
    }
  }
```



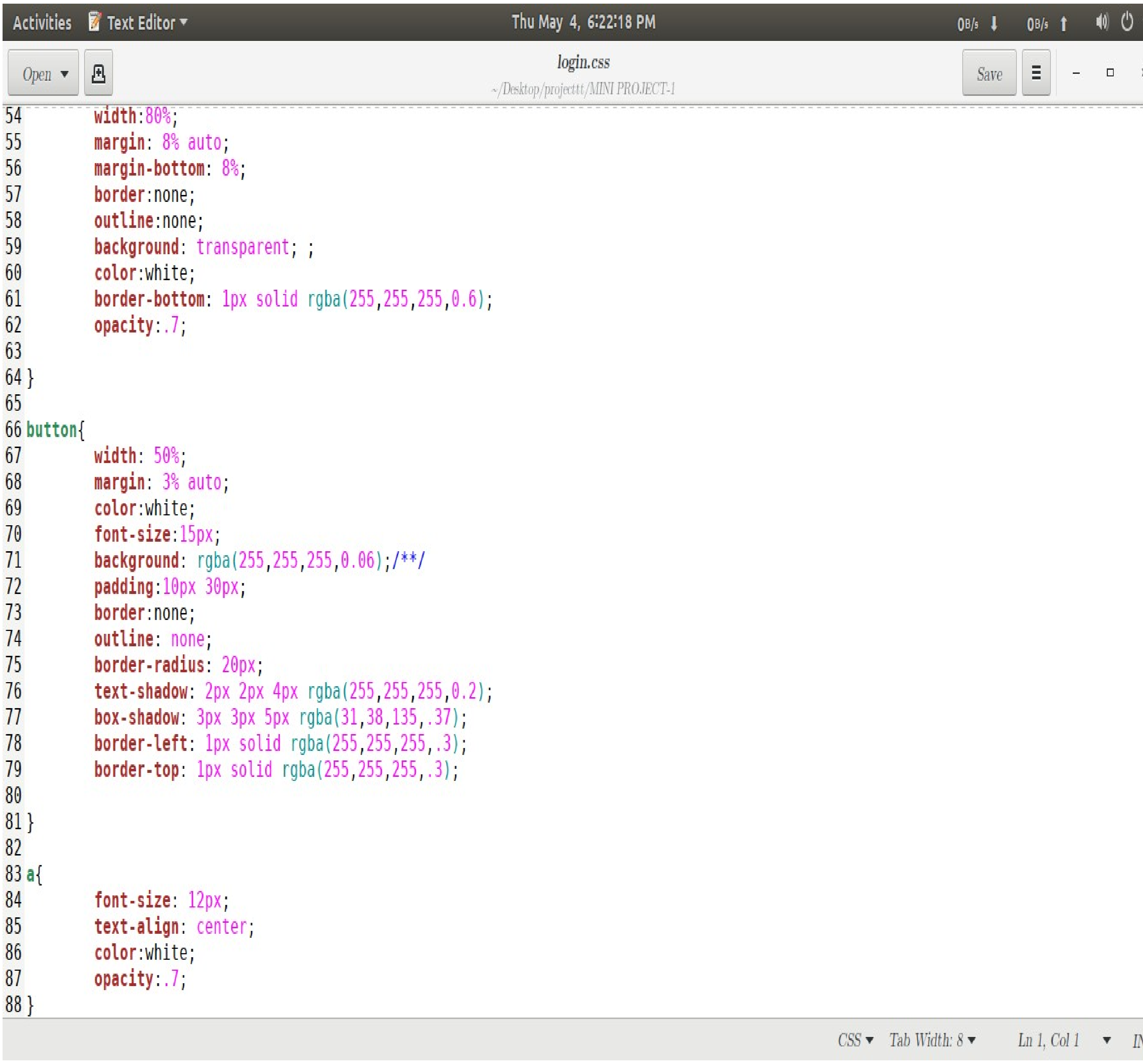
```


  if (!set) //on starting page
  {
    set_9x9('001700509573024106800501002700295018009400305652800007465080071000159004908007053');
  }



}





function clear_input()//for clearing input in onclick in input type in html
{
  for (var i = 0; i < 81; ++i)
  {
    document.getElementById('C' + i).value = '';
  }
}






function setPredefined(lvl) /*for Diffrent combinations in begsinner , expert etc--for buttons in html*/
{
  switch (lvl)
  {
    case 'beginner':
        a =
```

inst.html



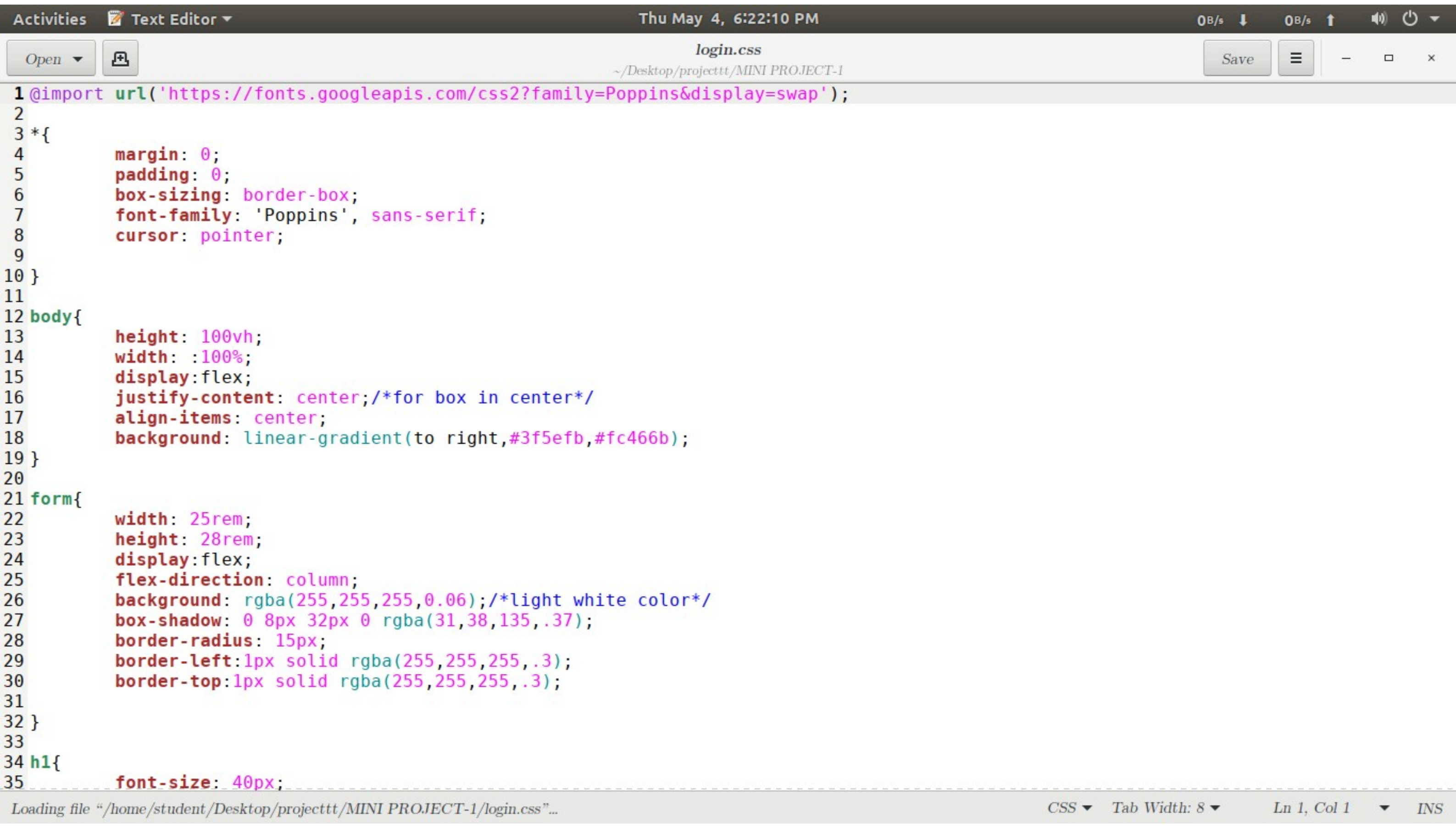
```css
@import url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');

*{
	margin: 0;
	padding: 0;
	box-sizing: border-box;
	font-family: 'Poppins', sans-serif;
	cursor: pointer;

}

body{
	height: 100vh;
	width: :100%;
	display:flex;
	justify-content: center;/*for box in center*/
	align-items: center;
	background: linear-gradient(to right,#3f5efb,#fc466b);
}

form{
	width: 25rem;
	height: 28rem;
	display:flex;
	flex-direction: column;
	background: rgba(255,255,255,0.06);/*light white color*/
	box-shadow: 0 8px 32px 0 rgba(31,38,135,.37);
	border-radius: 15px;
	border-left:1px solid rgba(255,255,255,.3);
	border-top:1px solid rgba(255,255,255,.3);

}

h1{
	font-size: 40px;
```

Loading file "/home/student/Desktop/projecttt/MINI PROJECT-1/login.css"...



```
	width:80%;
	margin: 8% auto;
	margin-bottom: 8%;
	border:none;
	outline:none;
	background: transparent; ;
	color:white;
	border-bottom: 1px solid rgba(255,255,255,0.6);
	opacity:.7;

}

button{
	width: 50%;
	margin: 3% auto;
	color:white;
	font-size:15px;
	background: rgba(255,255,255,0.06);/**/
	padding:10px 30px;
	border:none;
	outline: none;
	border-radius: 20px;
	text-shadow: 2px 2px 4px rgba(255,255,255,0.2);
	box-shadow: 3px 3px 5px rgba(31,38,135,.37);
	border-left: 1px solid rgba(255,255,255,.3);
	border-top: 1px solid rgba(255,255,255,.3);

}

a{
	font-size: 12px;
	text-align: center;
	color:white;
	opacity:.7;
}
```

about.html

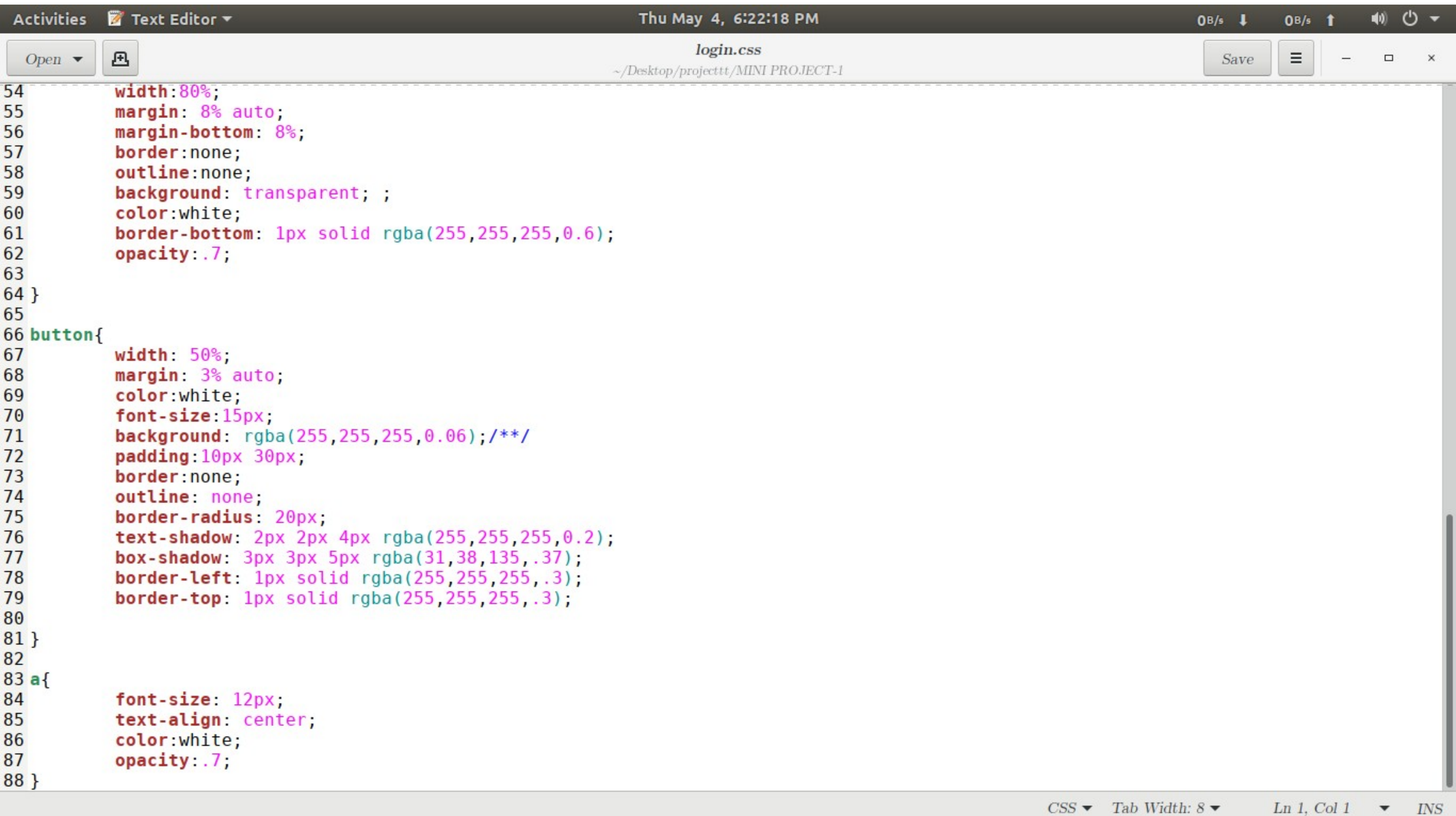


Activities Text Editor Thu May 4, 6:22:10 PM

Open login.css ~/Desktop/projecttt/MINI PROJECT-1 Save

```
@import url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');

*{
	margin: 0;
	padding: 0;
	box-sizing: border-box;
	font-family: 'Poppins', sans-serif;
	cursor: pointer;

}

body{
	height: 100vh;
	width: :100%;
	display:flex;
	justify-content: center;/*for box in center*/
	align-items: center;
	background: linear-gradient(to right,#3f5efb,#fc466b);
}

form{
	width: 25rem;
	height: 28rem;
	display:flex;
	flex-direction: column;
	background: rgba(255,255,255,0.06);/*light white color*/
	box-shadow: 0 8px 32px 0 rgba(31,38,135,.37);
	border-radius: 15px;
	border-left:1px solid rgba(255,255,255,.3);
	border-top:1px solid rgba(255,255,255,.3);

}

h1{
	font-size: 40px;
```

Loading file "/home/student/Desktop/projecttt/MINI PROJECT-1/login.css"... CSS Tab Width: 8 Ln 1, Col 1 INS



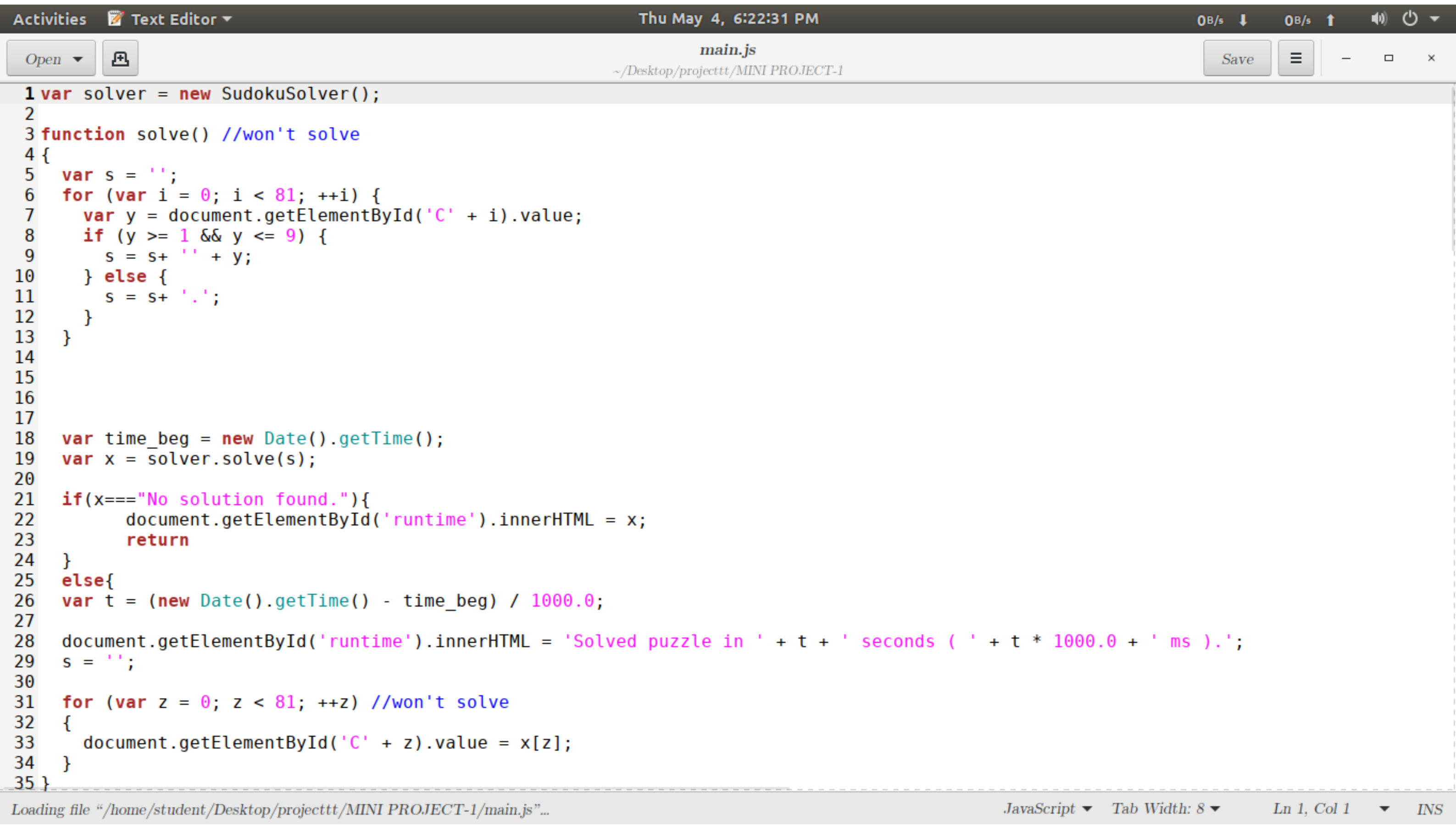
Activities Text Editor Thu May 4, 6:22:18 PM

Open login.css ~/Desktop/projecttt/MINI PROJECT-1 Save

```
	width:80%;
	margin: 8% auto;
	margin-bottom: 8%;
	border:none;
	outline:none;
	background: transparent; ;
	color:white;
	border-bottom: 1px solid rgba(255,255,255,0.6);
	opacity:.7;

}

button{
	width: 50%;
	margin: 3% auto;
	color:white;
	font-size:15px;
	background: rgba(255,255,255,0.06);/**/
	padding:10px 30px;
	border:none;
	outline: none;
	border-radius: 20px;
	text-shadow: 2px 2px 4px rgba(255,255,255,0.2);
	box-shadow: 3px 3px 5px rgba(31,38,135,.37);
	border-left: 1px solid rgba(255,255,255,.3);
	border-top: 1px solid rgba(255,255,255,.3);

}

a{
	font-size: 12px;
	text-align: center;
	color:white;
	opacity:.7;
}
```

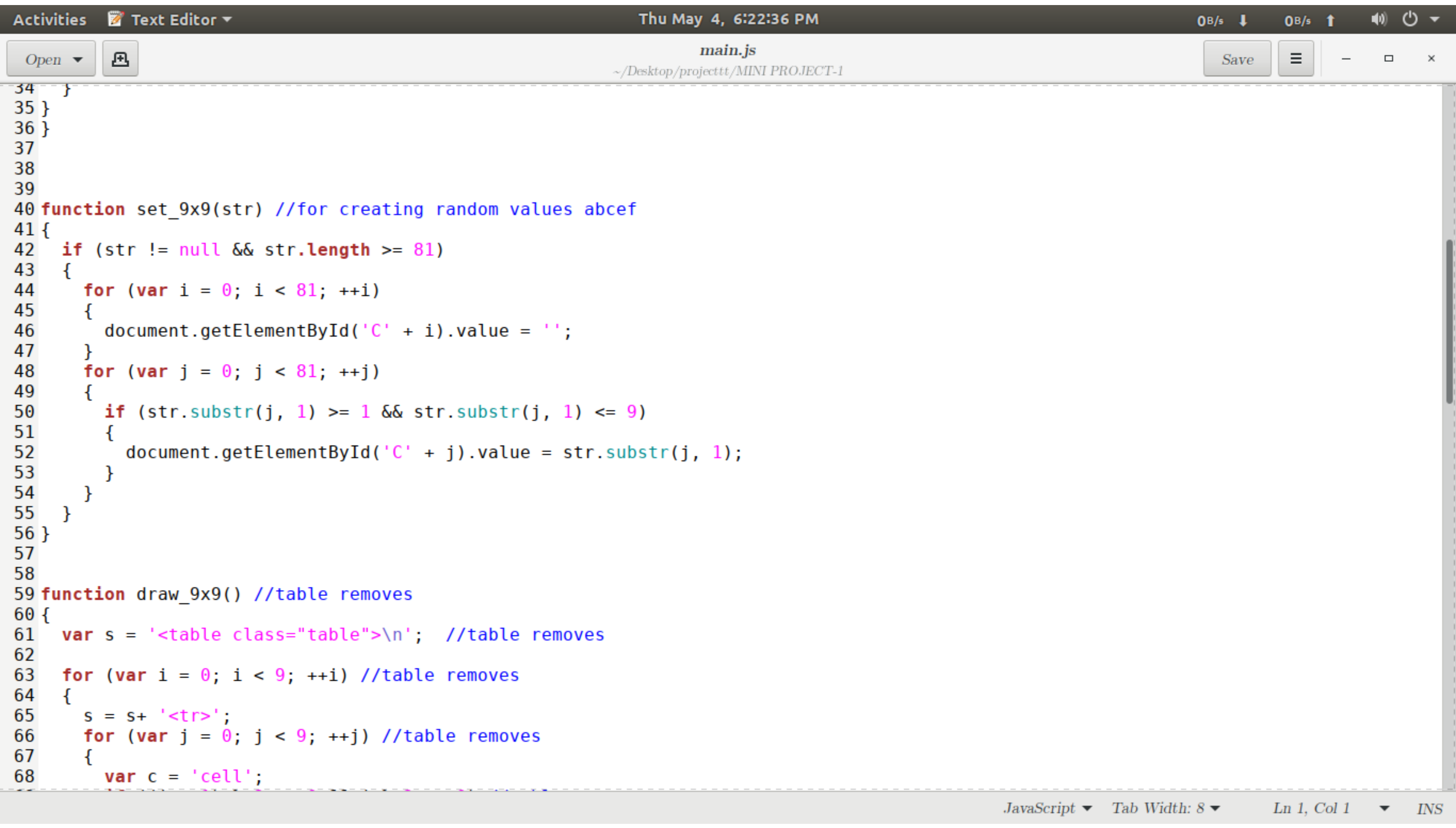
CSS Tab Width: 8 Ln 1, Col 1 INS

main.js

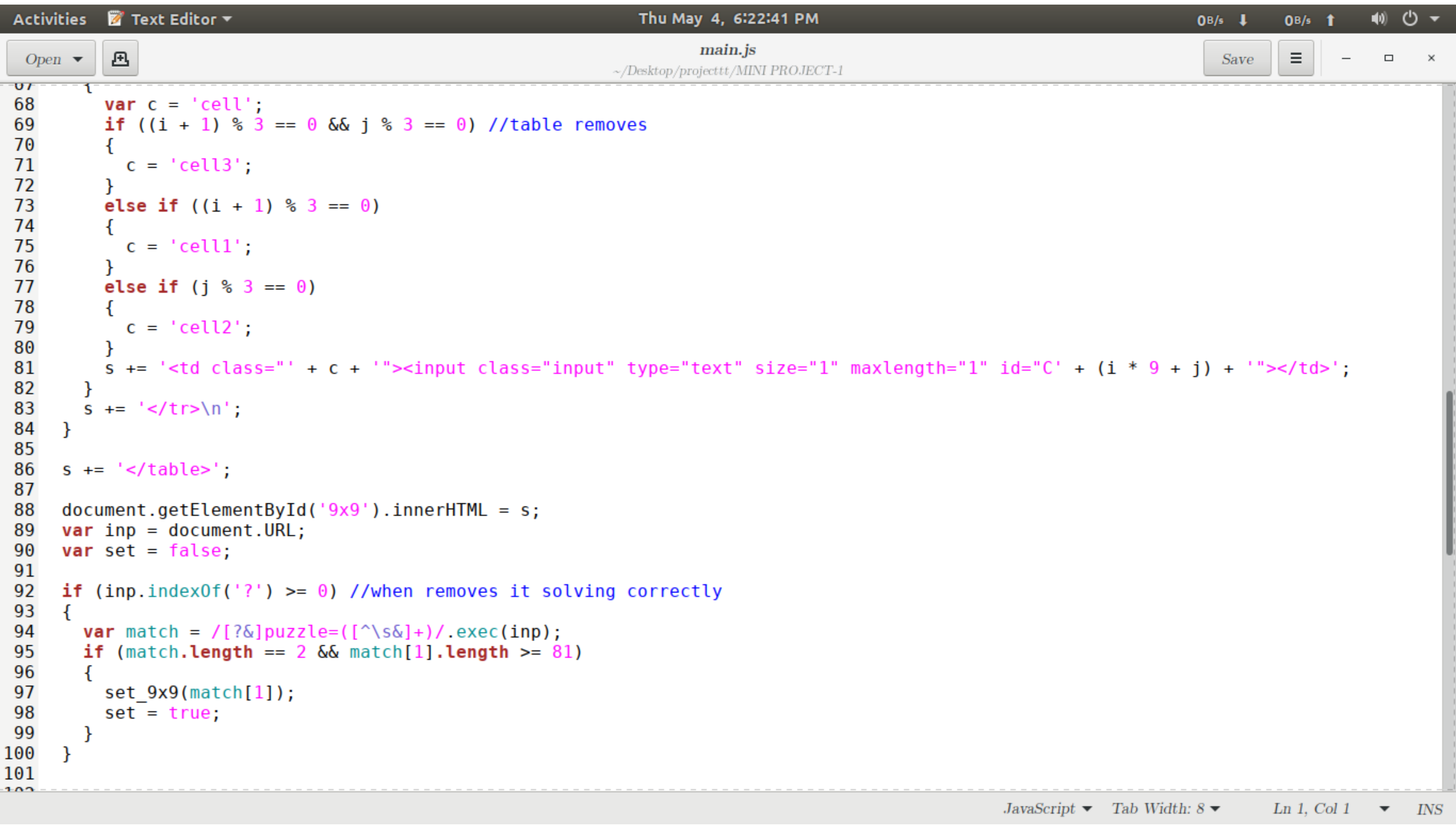


```
var solver = new SudokuSolver();

function solve() //won't solve
{
  var s = '';
  for (var i = 0; i < 81; ++i) {
    var y = document.getElementById('C' + i).value;
    if (y >= 1 && y <= 9) {
      s = s+ '' + y;
    } else {
      s = s+ '.';
    }
  }




  var time_beg = new Date().getTime();
  var x = solver.solve(s);

  if(x==="No solution found."){
        document.getElementById('runtime').innerHTML = x;
        return
  }
  else{
  var t = (new Date().getTime() - time_beg) / 1000.0;

  document.getElementById('runtime').innerHTML = 'Solved puzzle in ' + t + ' seconds ( ' + t * 1000.0 + ' ms ).';
  s = '';

  for (var z = 0; z < 81; ++z) //won't solve
  {
    document.getElementById('C' + z).value = x[z];
  }
}
```

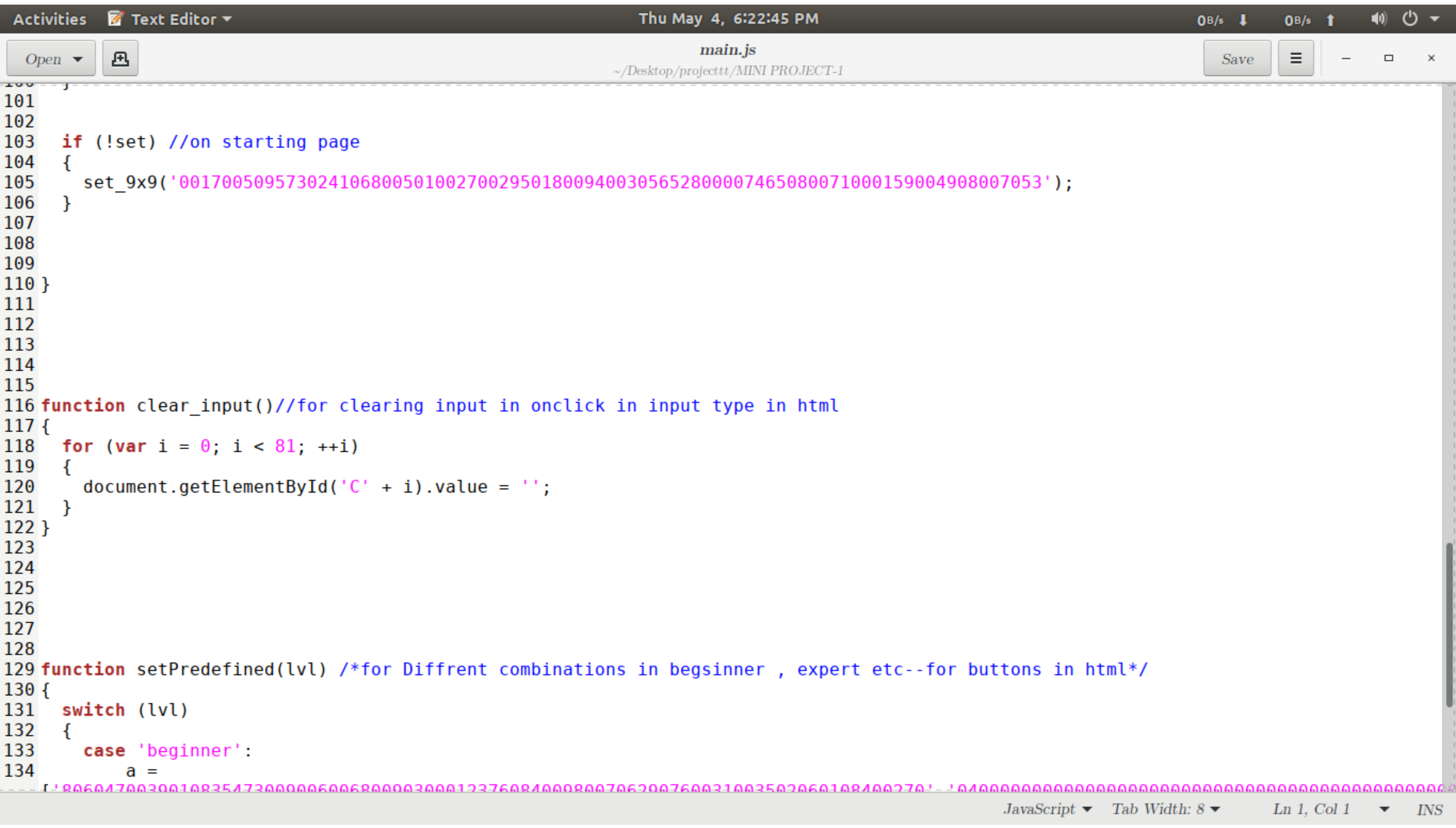
Loading file "/home/student/Desktop/projecttt/MINI PROJECT-1/main.js"...



```
  }
}
}



function set_9x9(str) //for creating random values abcef
{
  if (str != null && str.length >= 81)
  {
    for (var i = 0; i < 81; ++i)
    {
      document.getElementById('C' + i).value = '';
    }
    for (var j = 0; j < 81; ++j)
    {
      if (str.substr(j, 1) >= 1 && str.substr(j, 1) <= 9)
      {
        document.getElementById('C' + j).value = str.substr(j, 1);
      }
    }
  }
}


function draw_9x9() //table removes
{
  var s = '<table class="table">\n';  //table removes

  for (var i = 0; i < 9; ++i) //table removes
  {
    s = s+ '<tr>';
    for (var j = 0; j < 9; ++j) //table removes
    {
      var c = 'cell';
```

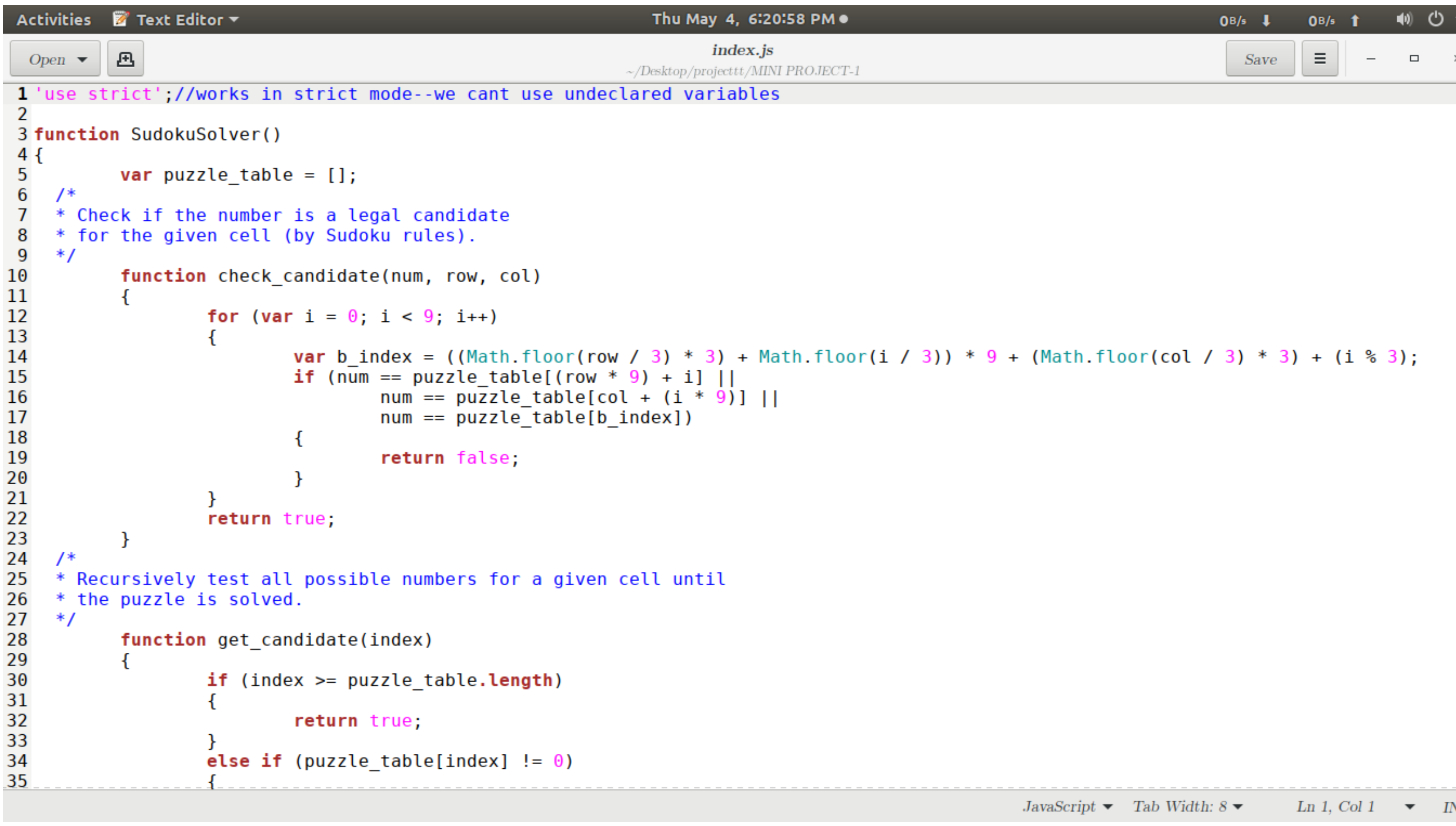


```
      var c = 'cell';
      if ((i + 1) % 3 == 0 && j % 3 == 0) //table removes
      {
        c = 'cell3';
      }
      else if ((i + 1) % 3 == 0)
      {
        c = 'cell1';
      }
      else if (j % 3 == 0)
      {
        c = 'cell2';
      }
      s += '<td class="' + c + '"><input class="input" type="text" size="1" maxlength="1" id="C' + (i * 9 + j) + '"></td>';
    }
    s += '</tr>\n';
  }

  s += '</table>';

  document.getElementById('9x9').innerHTML = s;
  var inp = document.URL;
  var set = false;

  if (inp.indexOf('?') >= 0) //when removes it solving correctly
  {
    var match = /[?&]puzzle=([^\s&]+)/.exec(inp);
    if (match.length == 2 && match[1].length >= 81)
    {
      set_9x9(match[1]);
      set = true;
    }
  }

```

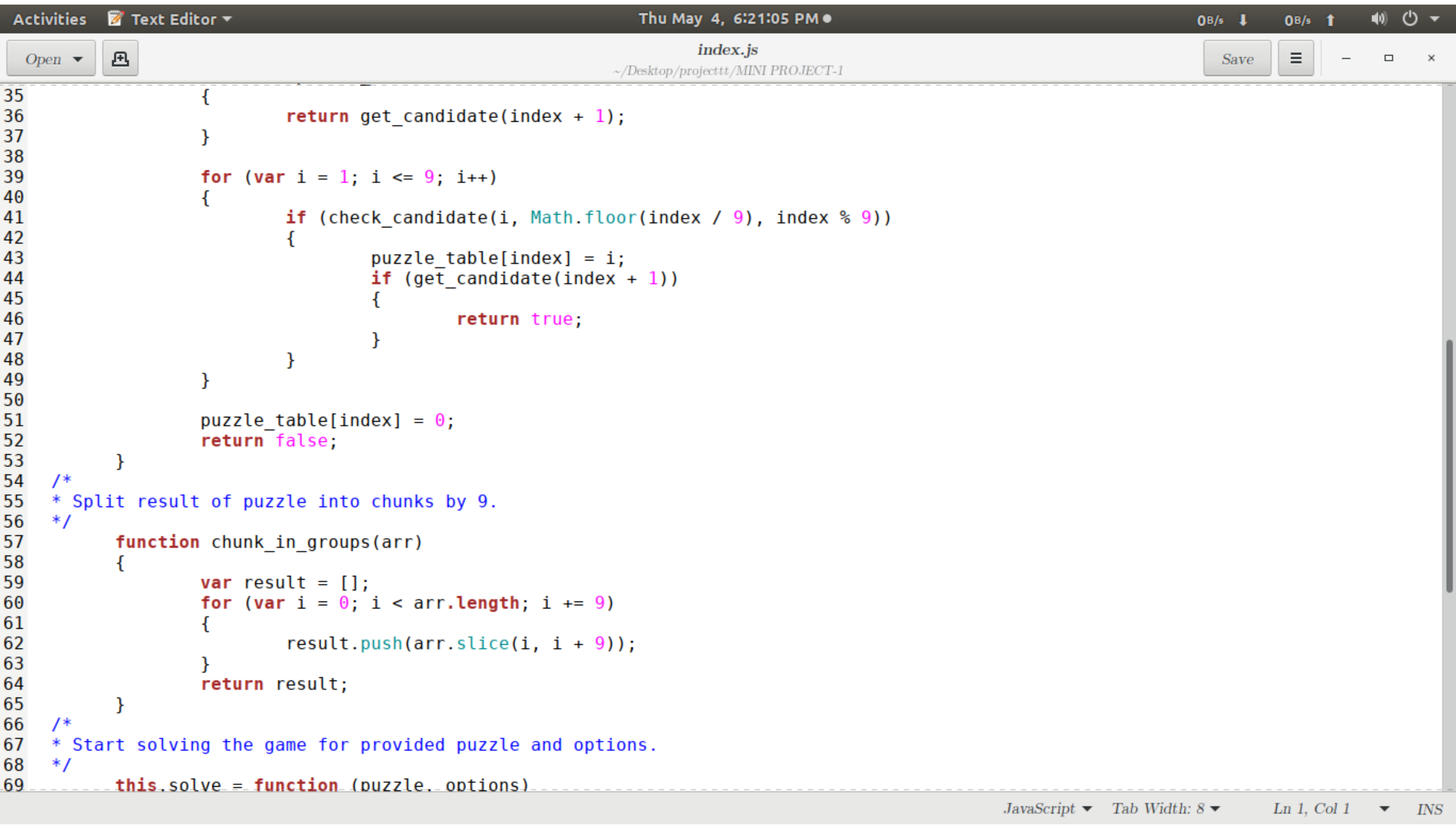


```


  if (!set) //on starting page
  {
    set_9x9('001700509573024106800501002700295018009400305652800007465080071000159004908007053');
  }



}





function clear_input()//for clearing input in onclick in input type in html
{
  for (var i = 0; i < 81; ++i)
  {
    document.getElementById('C' + i).value = '';
  }
}






function setPredefined(lvl) /*for Diffrent combinations in begsinner , expert etc--for buttons in html*/
{
  switch (lvl)
  {
    case 'beginner':
        a =
['806047003901083547300900600680090300012376084009800706290760031003502060108400270', '0400000000000000000000000000000000000000000000
```

index.js

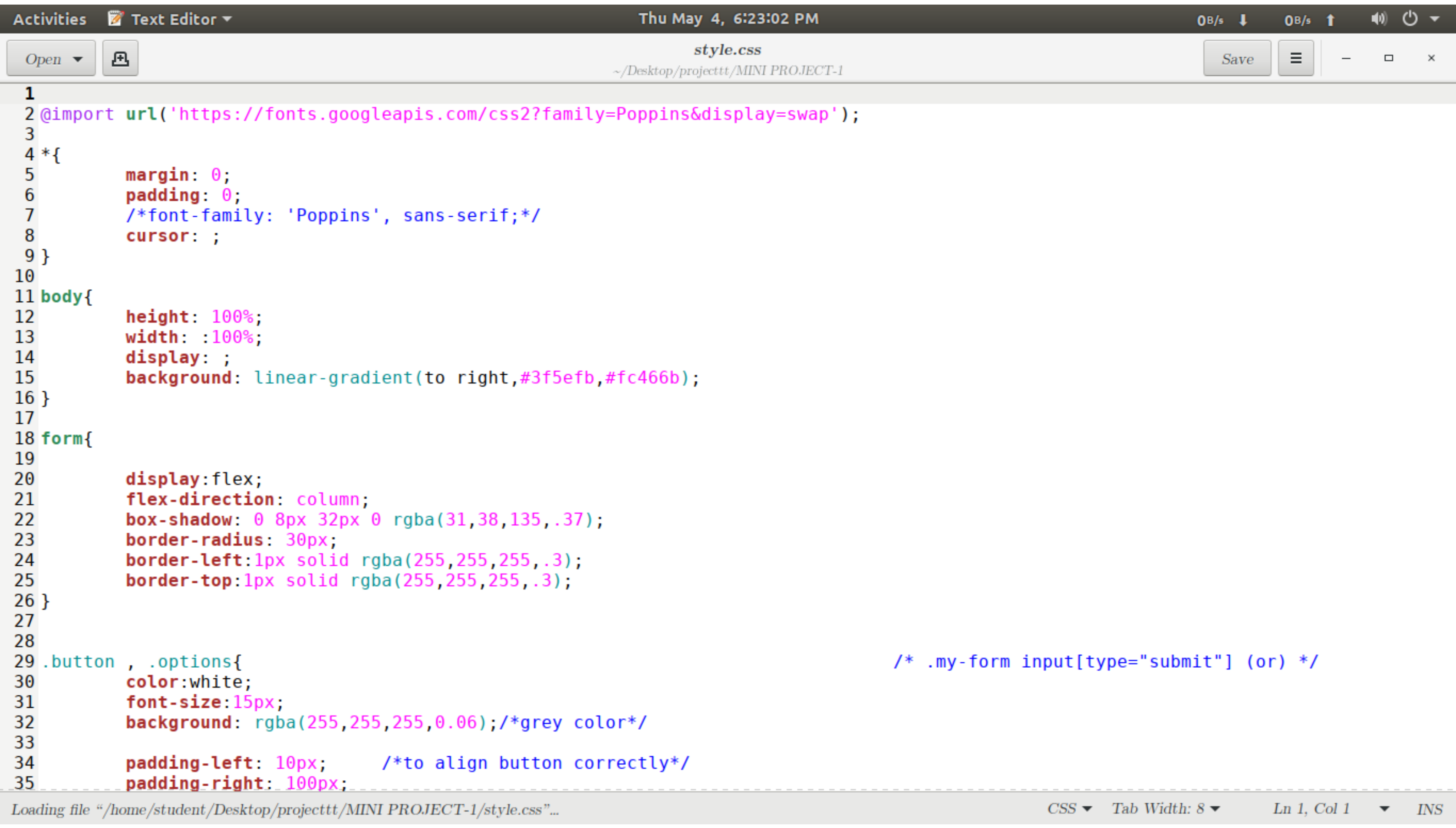


```javascript
'use strict';//works in strict mode--we cant use undeclared variables

function SudokuSolver()
{
        var puzzle_table = [];
  /*
  * Check if the number is a legal candidate
  * for the given cell (by Sudoku rules).
  */
        function check_candidate(num, row, col)
        {
                for (var i = 0; i < 9; i++)
                {
                        var b_index = ((Math.floor(row / 3) * 3) + Math.floor(i / 3)) * 9 + (Math.floor(col / 3) * 3) + (i % 3);
                        if (num == puzzle_table[(row * 9) + i] ||
                                num == puzzle_table[col + (i * 9)] ||
                                num == puzzle_table[b_index])
                        {
                                return false;
                        }
                }
                return true;
        }
  /*
  * Recursively test all possible numbers for a given cell until
  * the puzzle is solved.
  */
        function get_candidate(index)
        {
                if (index >= puzzle_table.length)
                {
                        return true;
                }
                else if (puzzle_table[index] != 0)
                {
```



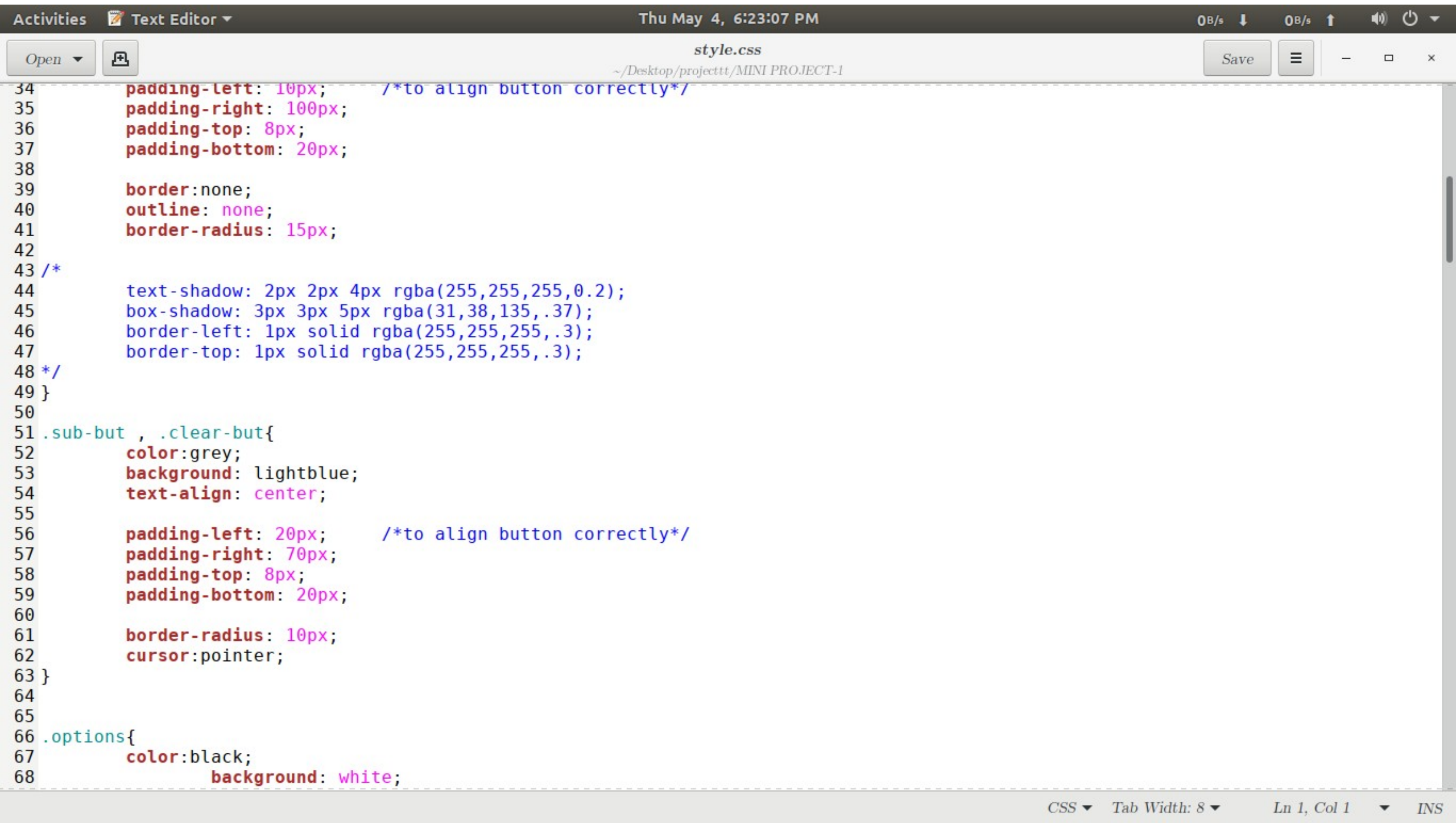
```javascript
                {
                        return get_candidate(index + 1);
                }

                for (var i = 1; i <= 9; i++)
                {
                        if (check_candidate(i, Math.floor(index / 9), index % 9))
                        {
                                puzzle_table[index] = i;
                                if (get_candidate(index + 1))
                                {
                                        return true;
                                }
                        }
                }

                puzzle_table[index] = 0;
                return false;
        }
  /*
  * Split result of puzzle into chunks by 9.
  */
        function chunk_in_groups(arr)
        {
                var result = [];
                for (var i = 0; i < arr.length; i += 9)
                {
                        result.push(arr.slice(i, i + 9));
                }
                return result;
        }
  /*
  * Start solving the game for provided puzzle and options.
  */
        this.solve = function (puzzle, options)
```

style.css

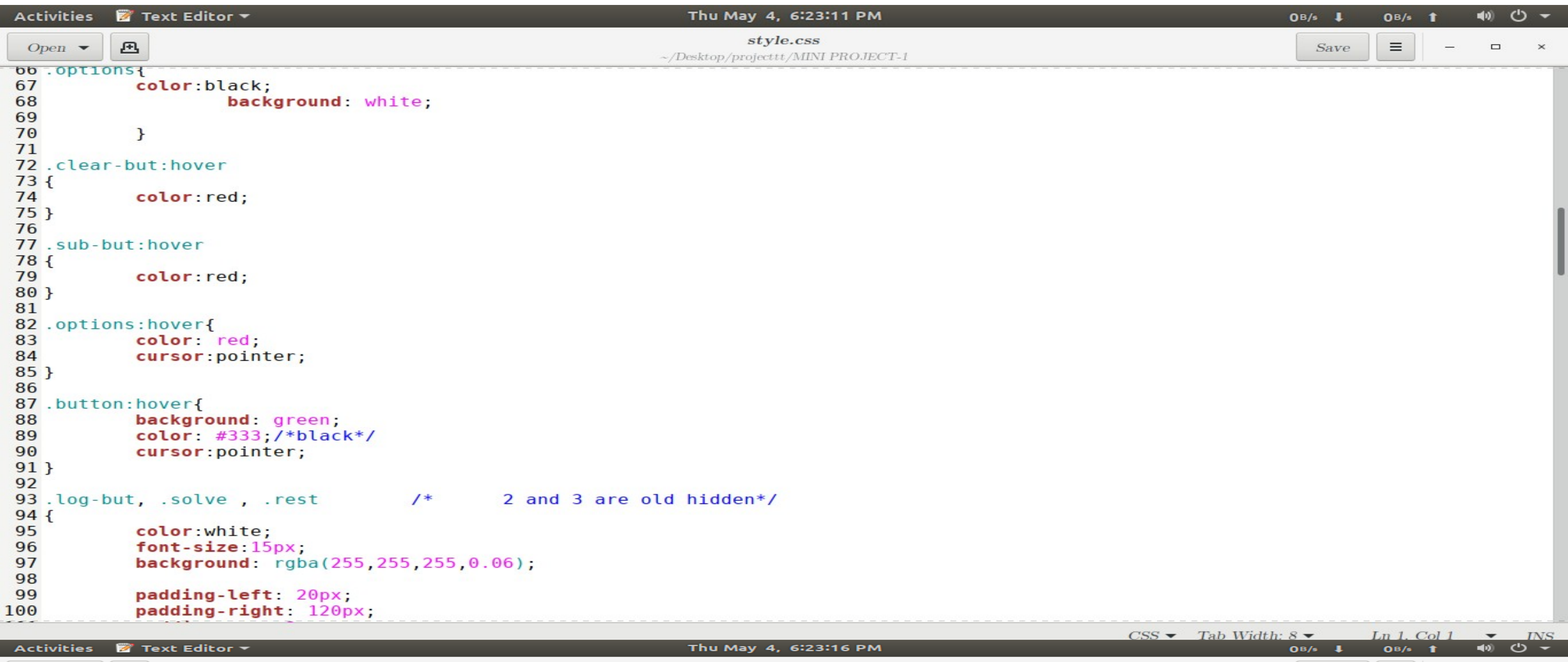


```css

@import url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');

*{
        margin: 0;
        padding: 0;
        /*font-family: 'Poppins', sans-serif;*/
        cursor: ;
}

body{
        height: 100%;
        width: :100%;
        display: ;
        background: linear-gradient(to right,#3f5efb,#fc466b);
}

form{

        display:flex;
        flex-direction: column;
        box-shadow: 0 8px 32px 0 rgba(31,38,135,.37);
        border-radius: 30px;
        border-left:1px solid rgba(255,255,255,.3);
        border-top:1px solid rgba(255,255,255,.3);
}


.button , .options{                                                             /* .my-form input[type="submit"] (or) */
        color:white;
        font-size:15px;
        background: rgba(255,255,255,0.06);/*grey color*/

        padding-left: 10px;     /*to align button correctly*/
        padding-right: 100px;
```

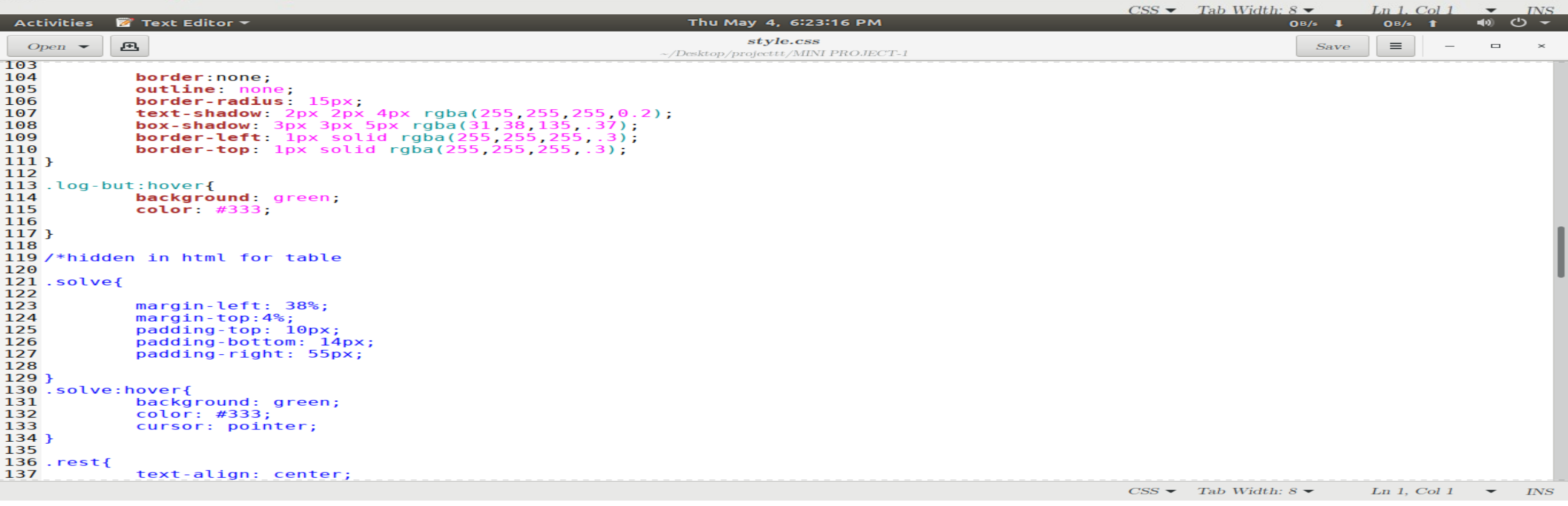
Loading file "/home/student/Desktop/projecttt/MINI PROJECT-1/style.css"...



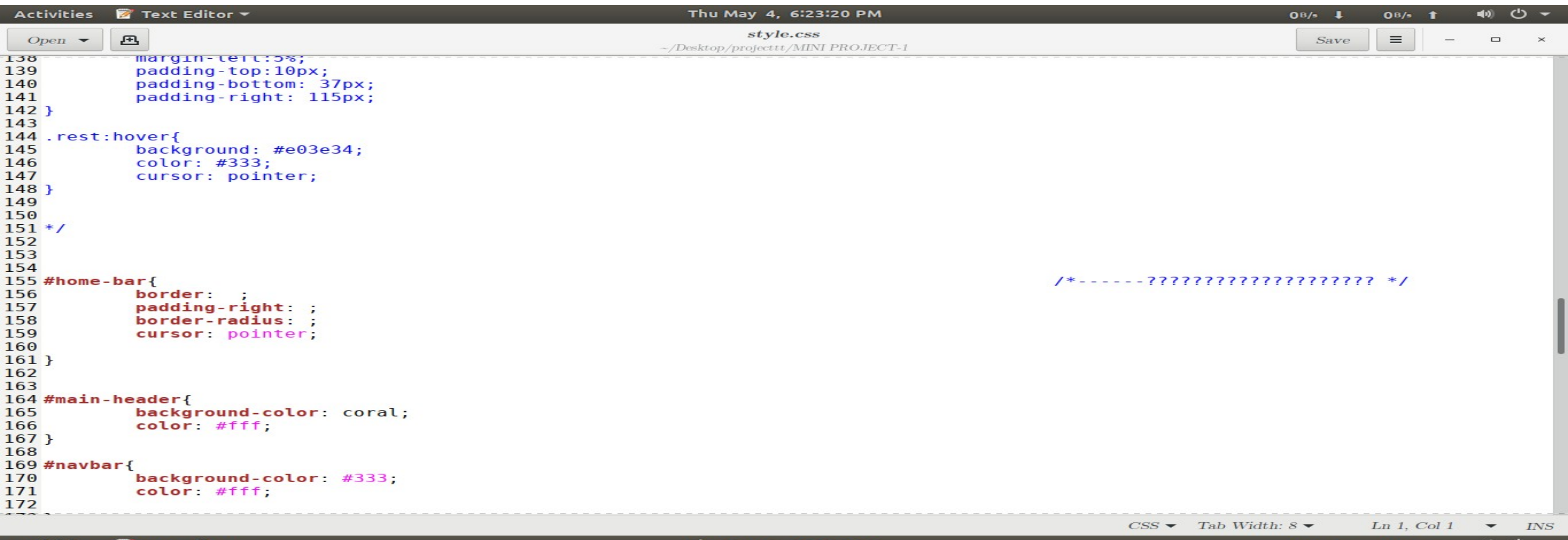
```css
        padding-left: 10px;     /*to align button correctly*/
        padding-right: 100px;
        padding-top: 8px;
        padding-bottom: 20px;

        border:none;
        outline: none;
        border-radius: 15px;

/*
        text-shadow: 2px 2px 4px rgba(255,255,255,0.2);
        box-shadow: 3px 3px 5px rgba(31,38,135,.37);
        border-left: 1px solid rgba(255,255,255,.3);
        border-top: 1px solid rgba(255,255,255,.3);
*/
}

.sub-but , .clear-but{
        color:grey;
        background: lightblue;
        text-align: center;

        padding-left: 20px;     /*to align button correctly*/
        padding-right: 70px;
        padding-top: 8px;
        padding-bottom: 20px;

        border-radius: 10px;
        cursor:pointer;
}


.options{
        color:black;
                background: white;
```



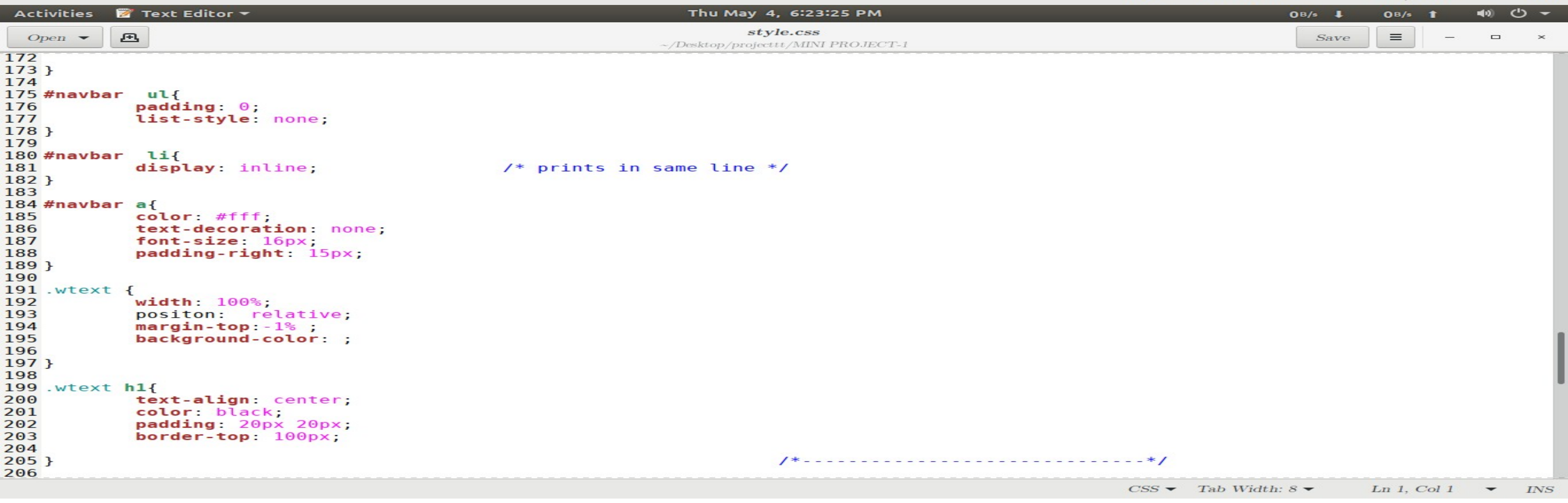
```
.options{
        color:black;
                background: white;

        }

.clear-but:hover
{
        color:red;
}

.sub-but:hover
{
        color:red;
}

.options:hover{
        color: red;
        cursor:pointer;
}

.button:hover{
        background: green;
        color: #333;/*black*/
        cursor:pointer;
}

.log-but, .solve , .rest        /*      2 and 3 are old hidden*/
{
        color:white;
        font-size:15px;
        background: rgba(255,255,255,0.06);

        padding-left: 20px;
        padding-right: 120px;
```



```
        border:none;
        outline: none;
        border-radius: 15px;
        text-shadow: 2px 2px 4px rgba(255,255,255,0.2);
        box-shadow: 3px 3px 5px rgba(31,38,135,.37);
        border-left: 1px solid rgba(255,255,255,.3);
        border-top: 1px solid rgba(255,255,255,.3);
}

.log-but:hover{
        background: green;
        color: #333;

}

/*hidden in html for table

.solve{

        margin-left: 38%;
        margin-top:4%;
        padding-top: 10px;
        padding-bottom: 14px;
        padding-right: 55px;

}
.solve:hover{
        background: green;
        color: #333;
        cursor: pointer;
}

.rest{
        text-align: center;
```

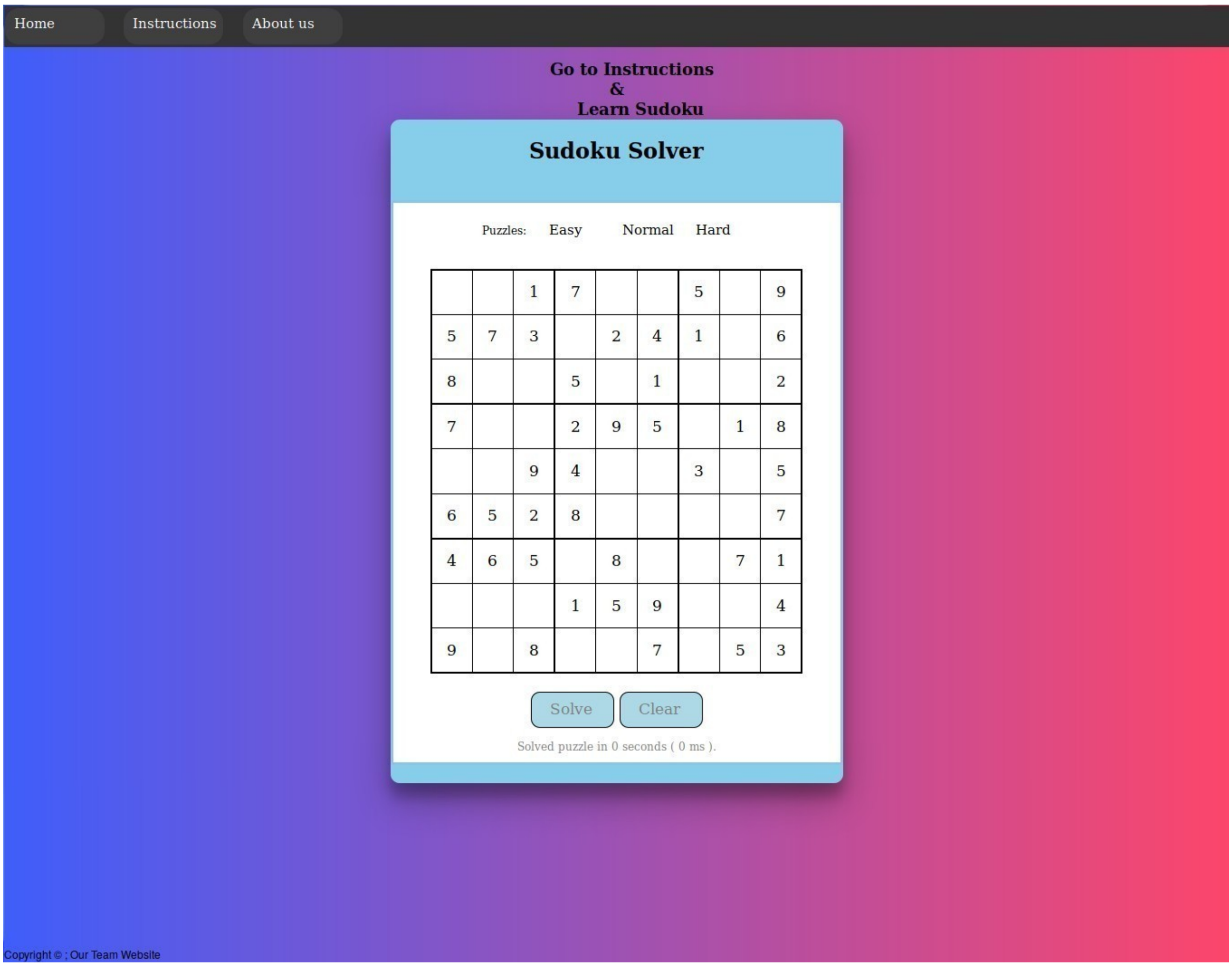


```
        margin-left:5%;
        padding-top:10px;
        padding-bottom: 37px;
        padding-right: 115px;
}

.rest:hover{
        background: #e03e34;
        color: #333;
        cursor: pointer;
}


*/



#home-bar{                                                                                      /*------???????????????????? */
        border:  ;
        padding-right: ;
        border-radius: ;
        cursor: pointer;

}


#main-header{
        background-color: coral;
        color: #fff;
}

#navbar{
        background-color: #333;
        color: #fff;

```



```

}

#navbar  ul{
        padding: 0;
        list-style: none;
}

#navbar  li{
        display: inline;                /* prints in same line */
}

#navbar a{
        color: #fff;
        text-decoration: none;
        font-size: 16px;
        padding-right: 15px;
}

.wtext {
        width: 100%;
        positon:  relative;
        margin-top:-1% ;
        background-color: ;

}

.wtext h1{
        text-align: center;
        color: black;
        padding: 20px 20px;
        border-top: 100px;

}                                                                   /*-------------------------------*/

```

OUTPUT :



Go to Instructions
&
Learn Sudoku

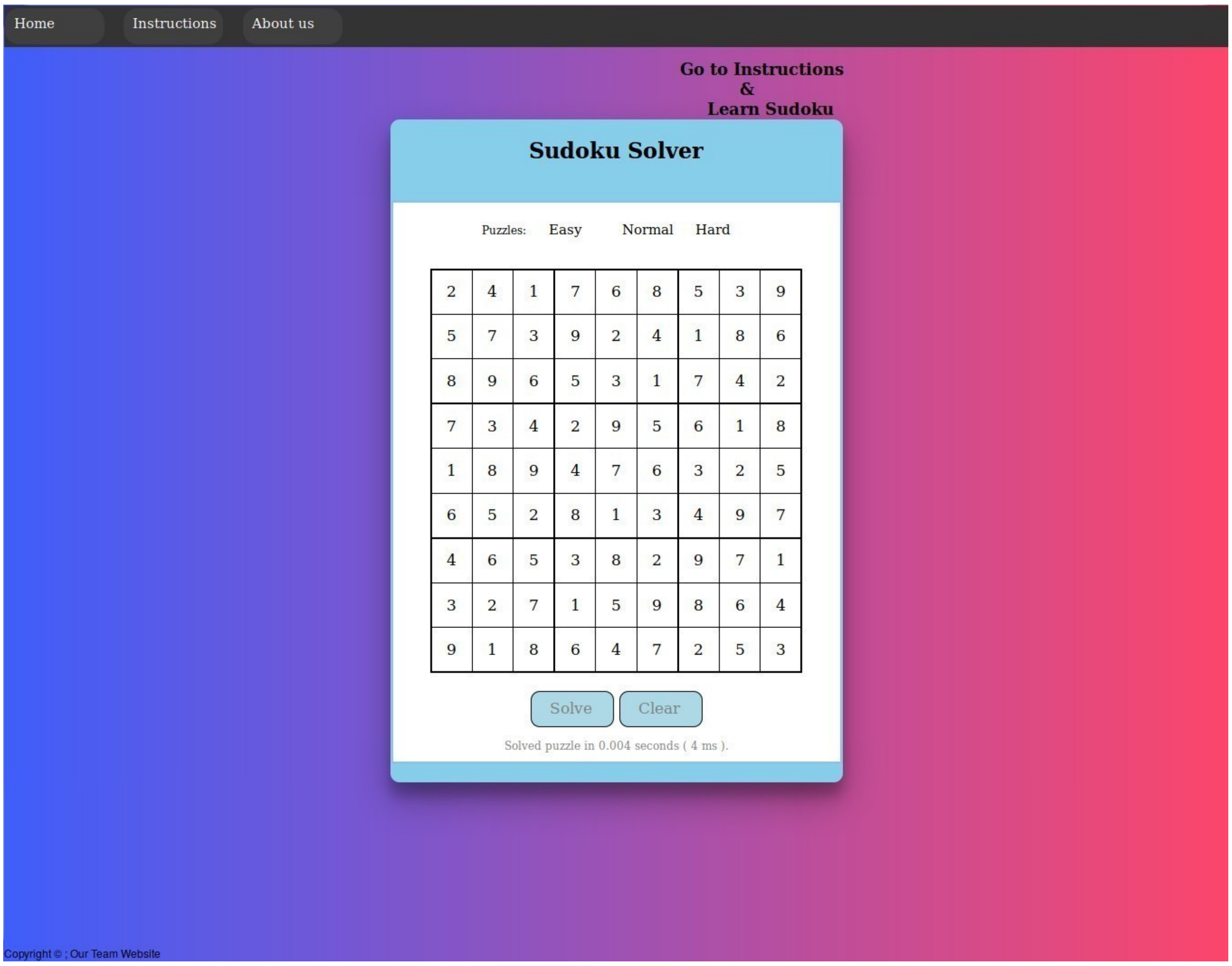
Sudoku Solver

Puzzles: Easy Normal Hard

		1	7			5		9
5	7	3		2	4	1		6
8			5		1			2
7			2	9	5		1	8
		9	4			3		5
6	5	2	8					7
4	6	5		8			7	1
			1	5	9			4
9		8			7		5	3

Solved puzzle in 0 seconds (0 ms).

Copyright © ; Our Team Website



Sudoku Solver

2	4	1	7	6	8	5	3	9
5	7	3	9	2	4	1	8	6
8	9	6	5	3	1	7	4	2
7	3	4	2	9	5	6	1	8
1	8	9	4	7	6	3	2	5
6	5	2	8	1	3	4	9	7
4	6	5	3	8	2	9	7	1
3	2	7	1	5	9	8	6	4
9	1	8	6	4	7	2	5	3

Solved puzzle in 0.004 seconds (4 ms).

Copyright © ; Our Team Website

Conclusions

This study has shown that the pencil-and-paper algorithm is a feasible method to solve any Sudoku puzzles. The algorithm is also an appropriate method to find a solution faster and more efficient compared to the brute force algorithm. The proposed algorithm is able to solve such puzzles with any level of difficulties in a short period of time (less than one second).

The testing results have revealed that the performance of the pencil-and-paper algorithm is better than the brute force algorithm with respect to the computing time to solve any puzzle.

The brute force algorithm seems to be a useful method to solve any Sudoku puzzles and it can guarantee to find at least one solution. However, this algorithm is not efficient because the level of difficulties is irrelevant to the algorithm. In other words, the algorithm does not adopt intelligent strategies to solve the puzzles. This algorithm checks all possible solutions to the puzzle until a valid solution is found which is a time consuming procedure resulting an inefficient solver. As it has already stated the main advantage of using the algorithm is the ability to solve any puzzles and a solution is certainly guaranteed.

Further research needs to be carried out in order to optimize the pencil-and-paper algorithm. A possible way could be implementing of other human strategies (x-wings, swordfish, etc.). Other alternatives might be to establish whether it is feasible to implement an algorithm based only on human strategies so that no other algorithm is involved in the pencil-and-paper algorithm and also make sure that these strategies can solve any puzzles with any level of difficulties.

References

1) Wikipedia [cited 2013 February 21], Web site: http://en.wikipedia.org/wiki/Sudoku

2) Home Of Logic Puzzles [cited 2013 February 22], Web Page: http://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/classic

3) J.F. Crook, A pencil and paper algorithm for solving Sudoku Puzzles, [Cited 2013 February 24], Winthrop University, Webpage: http://www.ams.org/notices/200904/tx090400460p.pdf

4) A.S. Showdhury, S. Skher Solving Sudoku with Boolean Algebra [Cited 2013 February 24], International Journal of Computer Applications, Peer-reviewed Research, Webpage: http://research.ijcaonline.org/volume52/number21/pxc3879024.pdf

5) N. Pillay, Finding Solutions to Sudoku Puzzles Using Human Intuitive Heuristics, South African Research Articles, Webpage: http://sacj.cs.uct.ac.za/index.php/sacj/article/viewArticle/111

6) Ch. Xu, W. Xu, The model and Algorithm to Estimate the Difficulty Levels of Sudoku Puzzles, Journal of Mathematics Research, 2009 Webpage: http://journal.ccsenet.org/index.php/jmr/article/viewFile/3732/3336

7) T. Davis, The Mathematics of Sudoku (http://www.geometer.org/index.html), Research Article, Webpage: http://share.dschola.it/castigliano/ips/Documentazione%20Progetto/Materiale%20Didattico/Matematica/1E/sudoku.pdf

8)The figures are taken from webpage: http://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/techniques

9) T. Kovacs, Artificial Intelligence through Search: Solving Sudoku Puzzles, Journal Papers, Webpage: http://www.cs.bris.ac.uk/Publications/Papers/2000948.pdf

10) Sudoku solver using brute force, visited in Mars 2013, https://github.com/olav/JavaSudokuSolver

11) The puzzle generator, visited in Mars 2013, websudoku.com/