SUDOKU brainteaser game application

By Daulet Kshibekov

1. Introduction.

The project is aimed at developing a Python-3 application of Sudoku brainteaser game that meets the following requirements:

- The application should generate a unique layout (9x9) of numbers from 1 to 9 with no duplicate numbers in each row, column and 3x3 boxes.
- These numbers should be filled in to 81 cells (text fields), but some of them should be hidden, so that a user could not see them.
- Blank cells with hidden numbers should be editable, while other cells with visible numbers should not be editable.
- The quantity of hidden numbers should depend on a difficulty level chosen by a user at the beginning of the game.
- Once game is started, a timer should start counting a time passed.
- User's goal would be to guess and fill in all the hidden numbers.
- 5 best results (in terms of shortest time) for each difficulty level should be saved to a file.
- Once user filled all blank cells with numbers and pressed "Check my solution" button, application should check a correctness.
- If the solution is correct, application should check if the user's time is shorter than those saved in a file.
- If it is shorter than at least one of 5 previously saved in the file records, or there are less than 5 records in the file, or there are no records yet, application should show an offering user to enter his/her name window, and should save a new record to a related to the current difficulty level file. The app should create such file if it does not exist.
- If user did not enter his/her name, record should be saved as "Anonymous" instead of the name.
- If the solution is incorrect (after user clicked the "Check my solution" button), the app should mark mistakes by red color and inform the user by a popup error window.
- Application should also validate each input if it is a number 1-9.
- Application should provide a possibility to choose a difficulty level.
- Application should have the following buttons:
  - Start new game.
  - Check solution.
- Application should have the following menu and sub menu items:
  - Game
    - New game
    - Check solution
    - Show correct solution
  - Best results.

- Application should show a table with the best results when the “Best results” menu item is chosen. If there are no results yet for the current difficulty level, the app should show an informing about that window.
- Application should stop current game and fill in all cells with the correct numbers when the “Show correct solution” menu item is chosen.
- Once the difficulty level in the combobox is changed, the application should start a new game with the related difficulty level.

2. Requirements.

The application needs the following imports:

- tkinter
- random
- time
- threading
- os.path

3. Description of the Python program.

The program consists of the following files:

- SudokuApp.py
- SudokuController.py
- LayoutGenerator.py
- NewRecord.py
- records_list_beginner.txt
- records_list_advanced.txt
- records_list_master.txt

SudocuApp.py consists of the following classes, methods and functions:

```
def vp_start_gui(): # application starting point.

def onClose(): # inner service function.

def destroy_Sudoku_brainteaser(): # inner service function.

class SudokuBrainteaser: # Defines application user interface.

    def buildCell(): # Defines tkinter Entry element as a cell for a number in layout.

    def buildRow(): # Defines a row of tkinter Entry elements (cells).
```

```
    def getCells(self): # Returns list of cell objects.

    def getDifficultyLevel(self): # Returns current difficulty level.
```

SudokuController.py consists of the following functions:

```
def showHOF(): ''' Depending on the difficulty level, loads the best results from a
related to the difficulty level file, and shows either window with the message
that there are no results yet or window with best results.'''

def processNewRecord(name): '''Saves new record to the related to the current difficulty
level file. Creates new file if it does not exist, adds record if there are less than 5 records,
deletes the worst result and adds new record if there are already 5 results in the file.'''

def checkRecord(): '''Opens related to the current difficulty level file and checks
if the time of the current game is shorter than those saved in the file.
Returns report in a tuple of three parameters:
- Boolean isNewRecord
- String name of the related to the current difficulty level file
- String operation type (new, append, rewrite).'''

def congratWinner(): '''Requests checkRecord() function to check if it is a new record.
Shows a window with congratulations and offering user to enter his name when a new
record is established.'''

def checkSolution(): '''Checks user's solution correctness. If solution is correct, will call
function congratWinner(). If is incorrect, will show error message.'''

def showSolution(): '''Stops game and fills in cells with correct numbers.'''

def timerStart(): '''Starts and shows a timer, counting a time passed since the game was
started.'''

def startNewGame(): '''Starts a new game. Requests a new layout, fills in cells with these
numbers,  hiding some of the numbers depending of the current difficulty level.  Starts a
timer.'''

def messageWindow(title, sortedLines): '''Shows a window with the best results.'''

def init(top, gui, *args, **kwargs): '''Initializer.'''
```

LayoutGenerator.py consists of the following classes and methods:

```
class LayoutGenerator: '''Generates and returns a unique layout (9x9) of numbers from 1 to
9  with no duplicate numbers in each row, column and 3x3 boxes.'''
```

```
    def __checkInCol(self, num, sudoku, n): '''Checks if the new number matches other
    numbers in the col.'''

    def __checkInRow(self, num, raw): '''Checks if the new number matches other
    numbers in the row.'''

    def __checkInBox(self, num, sudoku, n): '''Checks if the new number matches other
    numbers in the 3x3 box.'''

    def __getRow(self, sudoku): '''Generates a row.'''

    def generateLayout(self): '''Main method of a class. Generates and returns layout."
```

NewRecord.py consists of the following classes and methods:

```
class New_record: '''Defines new record window user interface.'''

    def processNewRec(self): '''Sends winner's name to
    SudokuController.processNewRecord() and disables the window.'''
```

records_list_beginner.txt, records_list_advanced.txt, and records_list_master.txt contain top 5 records for each difficulty level. Will be created automatically by the program if do not exist.

The application's entry point is *SudokuApp.py* file. *vp_start_gui()* function creates user interface by creating *SudokuBrainteaser* class instance, initializes *SudokuController.ry* module by calling *SudokuController.init(root, top)* function, and starts a new game by calling *SudokuController.startNewGame()* function.

SudokuController.startNewGame() function gets a new layout of numbers by calling *generateLayout()* method of the *LayoutGenerator* class instance. Then it marks a dependent on the current difficulty level quantity of numbers as hidden. It chooses which exactly numbers should be hidden with the help of *Random* module. Then it fills the remaining numbers into accorded cells and starts a timer by calling *Timer*'s instance's *start()* method.

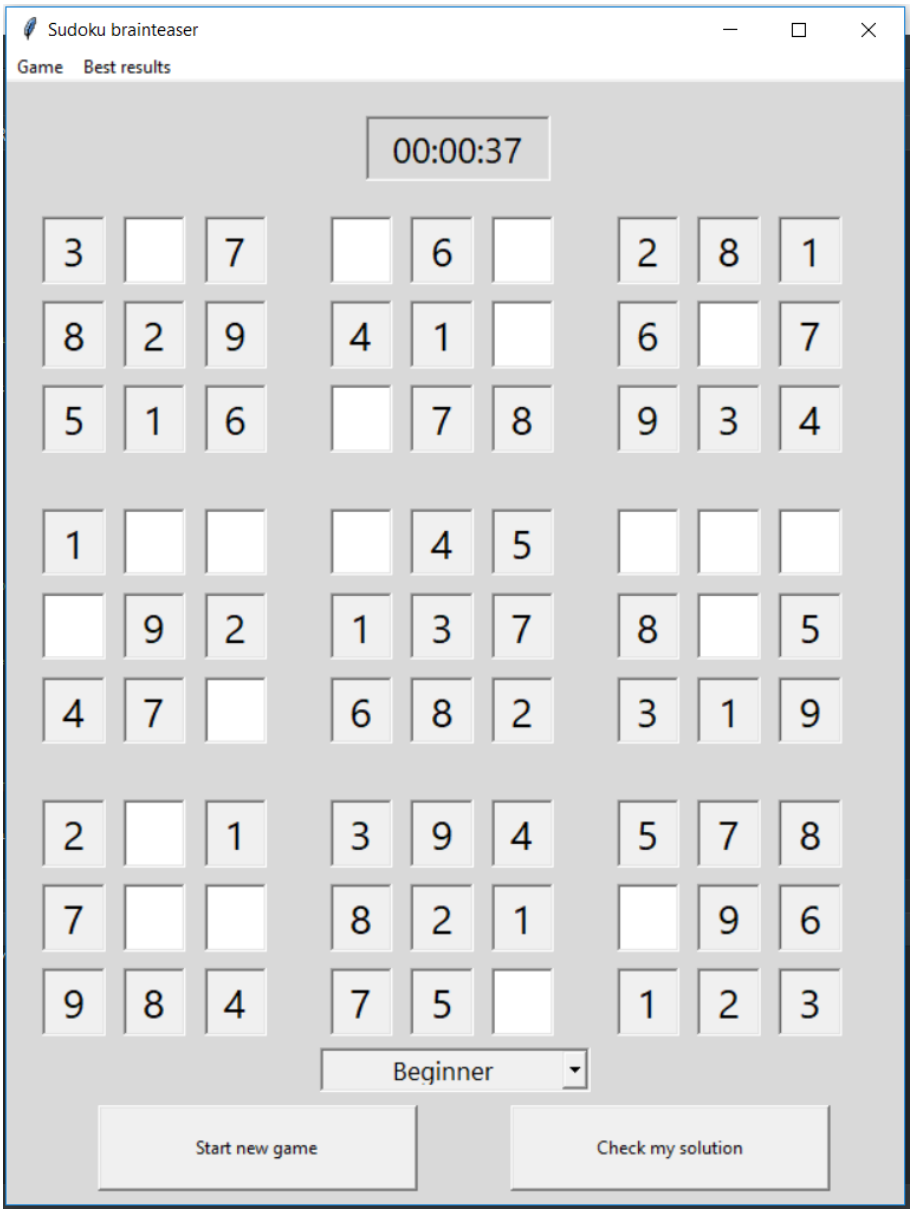
Once the user filled in all empty cells, he/she clicks the "Check my solution" button, which calls *SudokuController.checkSolution()* function. This function checks the user's solution correctness, and if it is correct it will call *congratWinner()* function. If the solution is incorrect, it will show the regarded message.

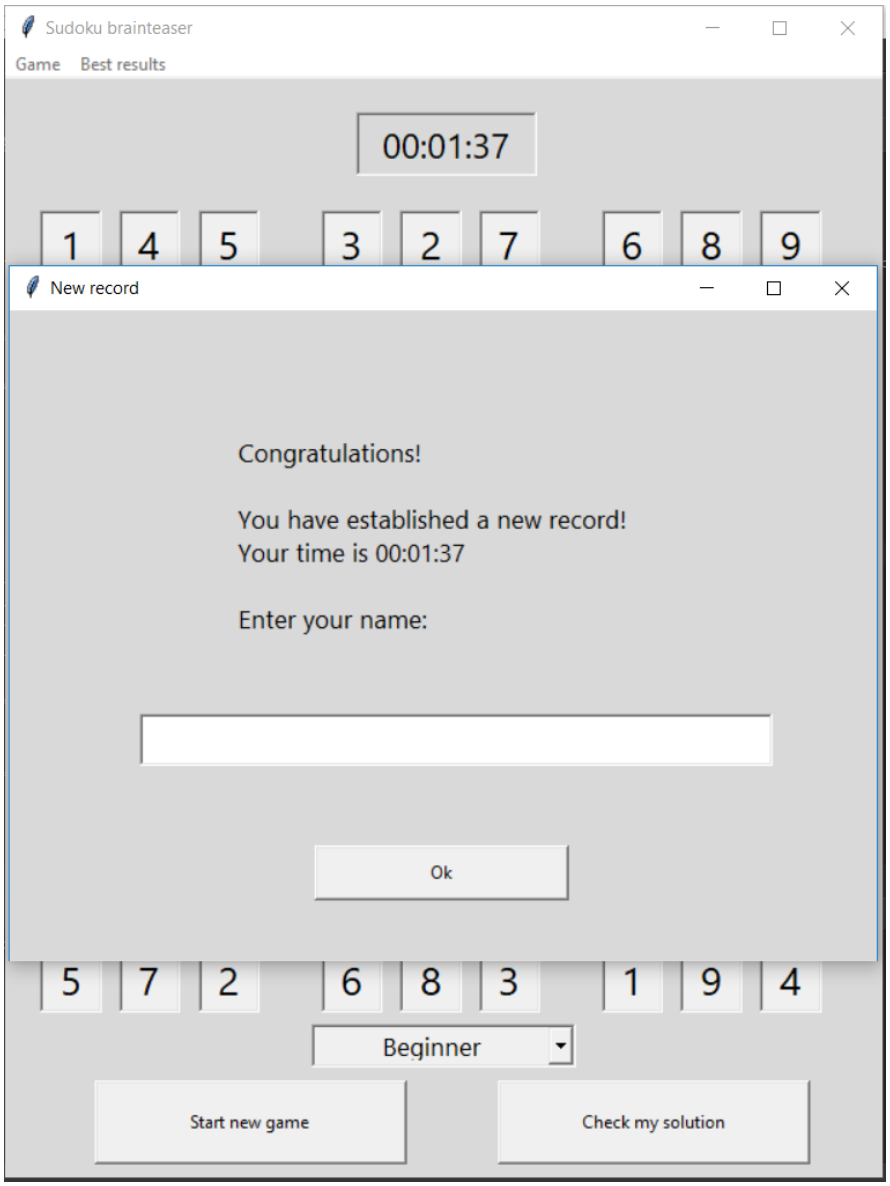
congratWinner() function checks if it is a new record by calling *checkRecord()* function, which compares the current time (*sec*) with those saved in the accorded to the current difficulty level file. If it is a new record, *congratWinner()* will create a NewRecord class instance by calling *NewRecord.create_New_record()* function, which will show a congratulation window prompting the user to enter his/her name, and will save the new record to the file. If it is not a new record, it will show congratulation window with the time result.

Once the user clicked the “Start new game” button, it calls *startNewGame()* function which starts a new game.

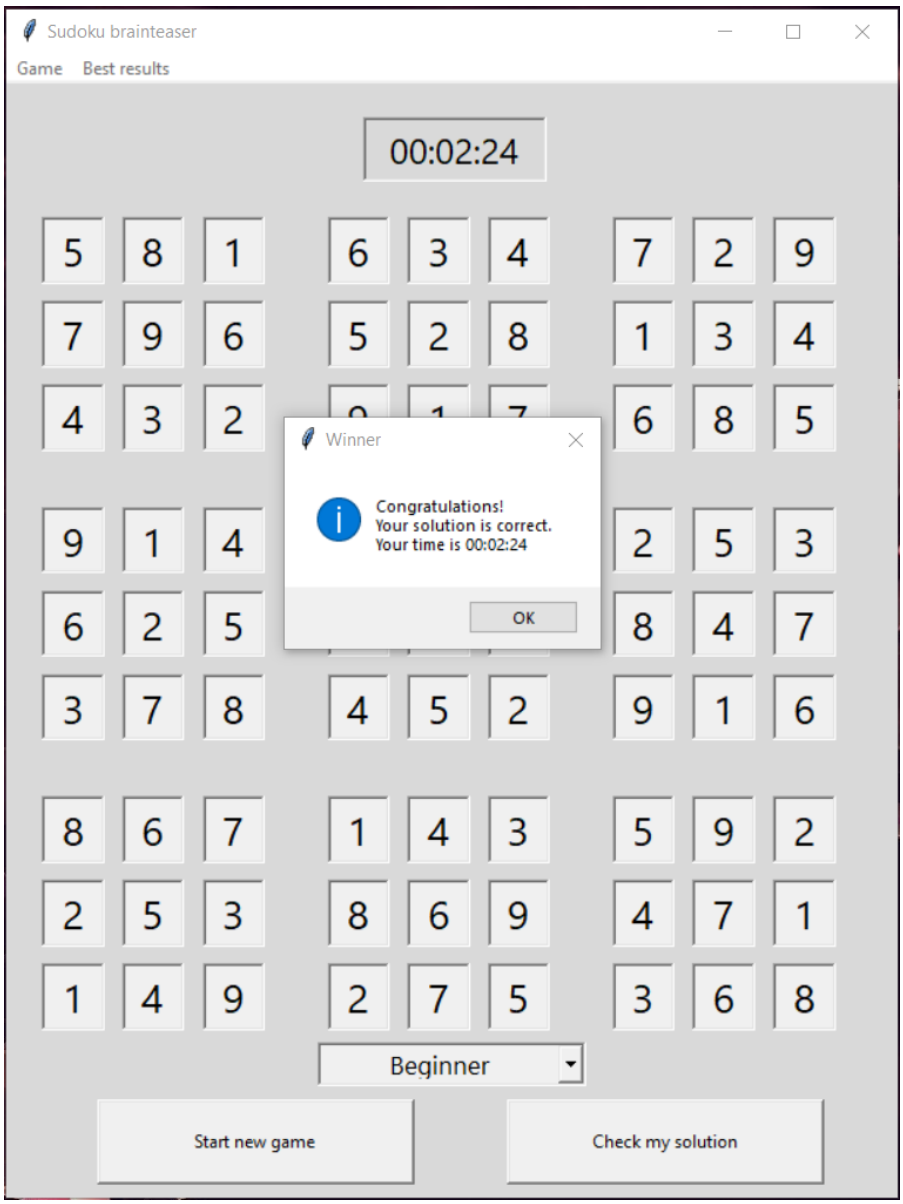
Once the user chosen “Show correct solution” menu item, it calls *showSolution()* function, which stops current game and fills in cells with correct numbers, making all cells not editable.

4. Screenshots of the program output.

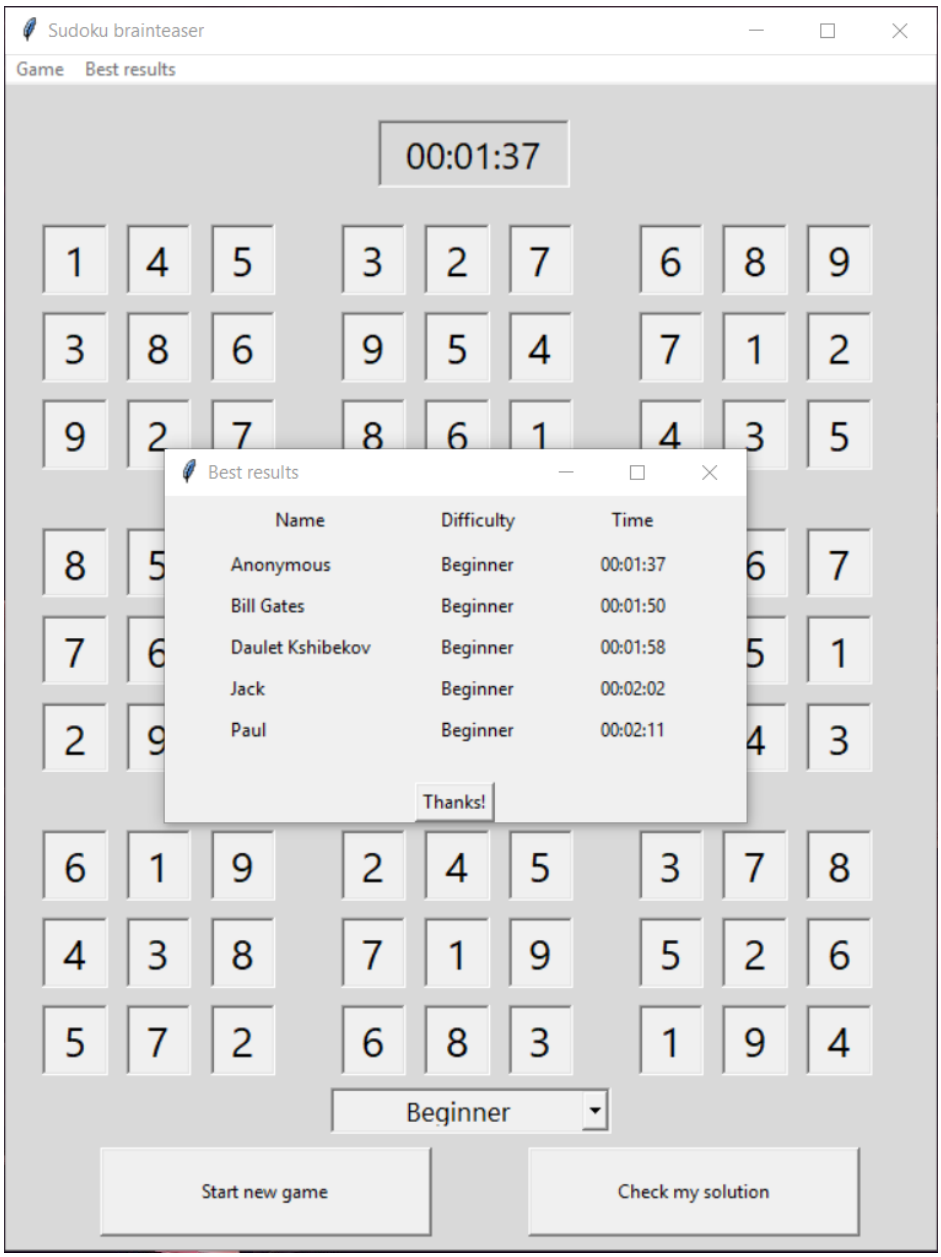


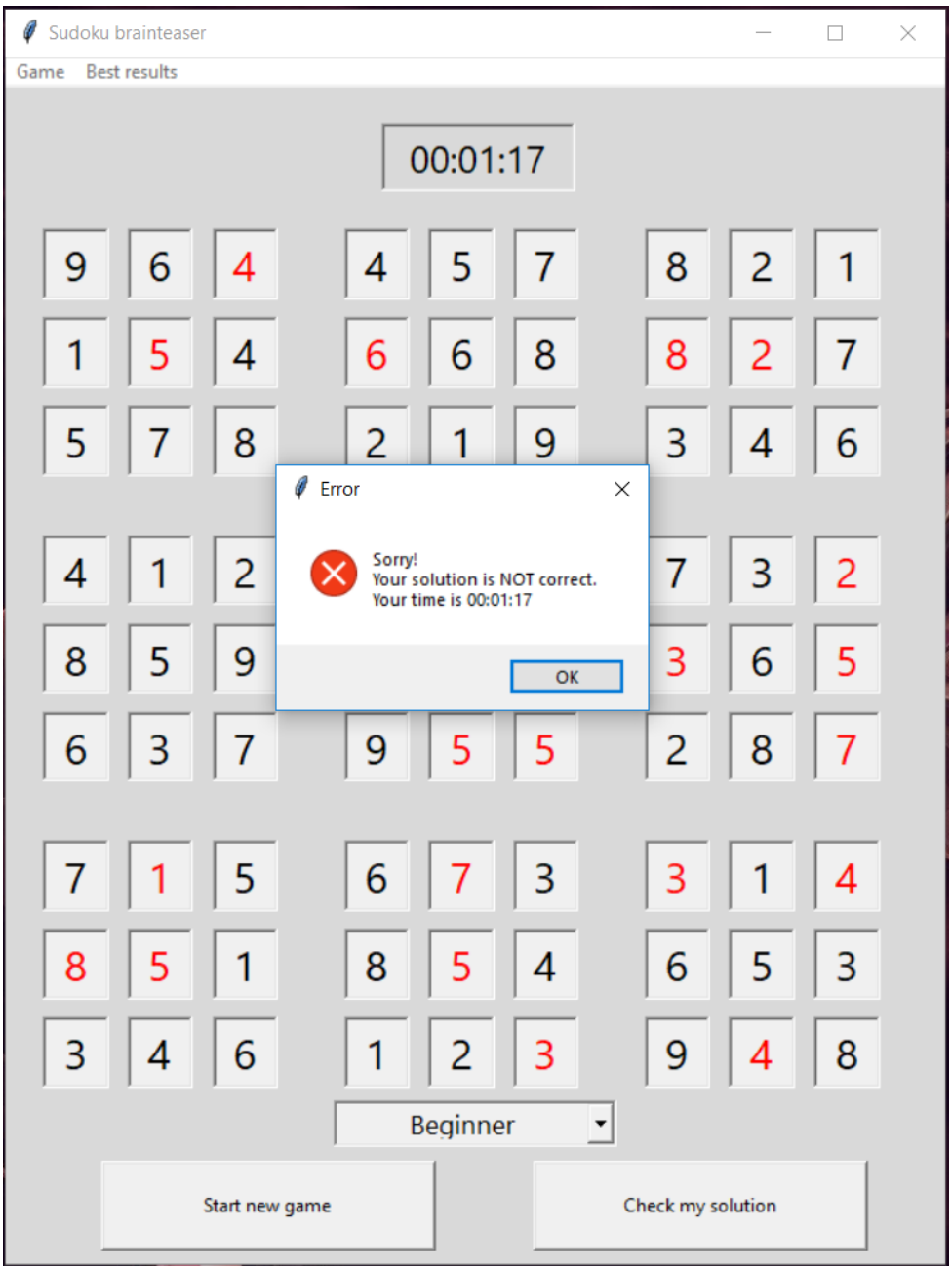


The player is a winner who established a new record.



Just a winner (correct solution). Not a new record.


Sudoku brainteaser
Game
Best results
00:01:37
1 4 5 3 2 7 6 8 9
3 8 6 9 5 4 7 1 2
9 2 7 8 6 1 4 3 5
8 5 6 7
7 6 5 1
2 9 4 3
6 1 9 2 4 5 3 7 8
4 3 8 7 1 9 5 2 6
5 7 2 6 8 3 1 9 4
Beginner
Start new game
Check my solution
Best results
Name
Difficulty
Time
Anonymous
Beginner
00:01:37
Bill Gates
Beginner
00:01:50
Daulet Kshibekov
Beginner
00:01:58
Jack
Beginner
00:02:02
Paul
Beginner
00:02:11
Thanks!



5. Conclusion.

The application is created in full conformity with the requirements listed in p.1 of the present document. It is stable and runs without errors.

This application is very useful for everyone as it can help to improve one's brain activity, release stress, get rested, and it is just good for entertainment.