BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	Alvin	Nathaniel	2440042430

Course Code : COMP6056 **Course Name** : Program Design Methods

Class : L1AC **Name of Lecturer(s)** : Jude Joseph Lamug Martinez

Major : Computer Science

Title of Assignment : Sudoku Solver

Type of Assignment : Final Project

Submission Pattern

Due Date : 13-01-2021 **Submission Date** :

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Nathaniel Alvin

"Sudoku Solver"

Name : Nathaniel Alvin

ID : 2440042430

I. Program Description

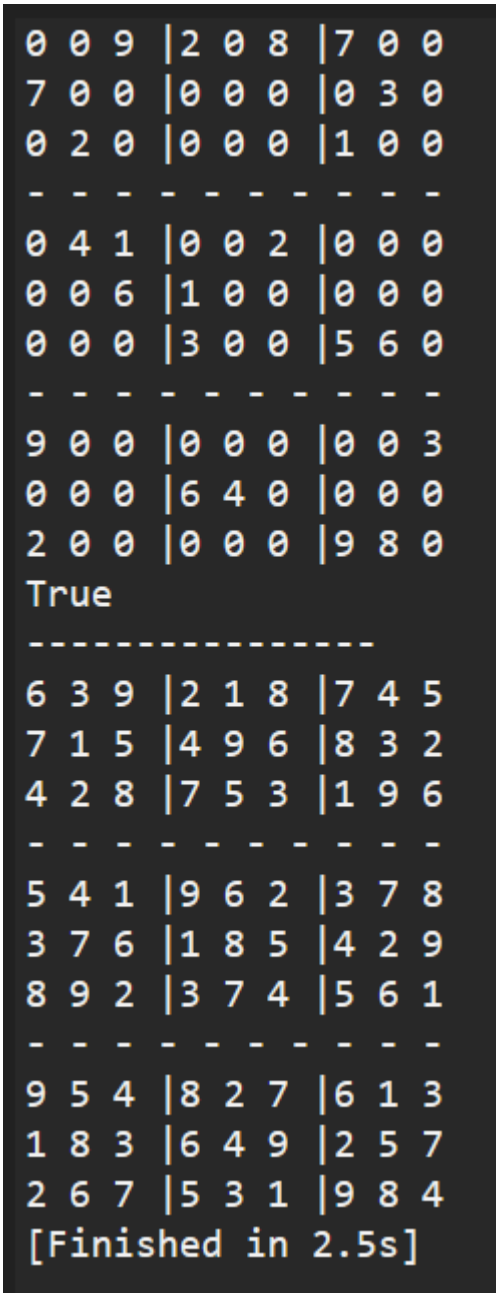
This application allows people to solve sudoku puzzles within seconds. It uses backtracking algorithm to deal with the problem. For now, we have to manually input the puzzle to the code.

II. How the program works

```
# This is the sudoku puzzle
grid = [
    [0, 0, 9, 2, 0, 8, 7, 0, 0],
    [7, 0, 0, 0, 0, 0, 0, 3, 0],
    [0, 2, 0, 0, 0, 0, 1, 0, 0],
    [0, 4, 1, 0, 0, 2, 0, 0, 0],
    [0, 0, 6, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 3, 0, 0, 5, 6, 0],
    [9, 0, 0, 0, 0, 0, 0, 0, 3],
    [0, 0, 0, 6, 4, 0, 0, 0, 0],
    [2, 0, 0, 0, 0, 0, 9, 8, 0]
]

```

As said before, the puzzle needs to be manually inputted to the code. 0 means blank space.



Then, run the program. The unsolved puzzle will be printed on top and the solution under it. True means the puzzle is solvable, and False means it is not.

III. Lessons that have been learned

Python is a new language for me therefore there is much room for improvement. This project teaches me a new algorithm called backtracking and how to implement it using python. It has been a fun project as every trial and error teaches me new things. Not only the algorithm, but the function to print the puzzle is also a bit tricky for me and it definitely improves my problem-solving skill.

IV. Code explanation

```
23  def insert_number(bo, row, col, n):  # test and imput a number to the grid
24      row_values = bo[row]
25      if n in row_values:  # check if there is number n on the row
26          return False
27      col_values = []
28      for i in range(9):  # check if there is number n on the column
29          col_values.append(bo[i][col])
30      if n in col_values:
31          return False
32
33      r0 = (row // 3) * 3  # create row for 3x3 grid
34      c0 = (col // 3) * 3  # create column for 3x3 grid
35
36      for i in range(0, 3):
37          for j in range(0, 3):
38              # accessing all the tiles in 3x3 grid + check
39              if bo[r0 + i][c0 + j] == n and (i, j) != (row, col):
40                  return False
41      return True
```

This insert_number function is to check whether it is possible to input a number into the blank space. It will return True if the number is possible and False if not possible. This function takes 4 arguments. Bo which is short for board, row and col or the position of the blank space and n which is the number we want to input. First, it will check if the number inputted is within the row and column. By using the in statement, this is very possible. For the column however, I need to append the values in the row into a list to make it easier. In a sudoku puzzle, there is a 3x3 grid or so-called sub-grid. Besides the row and column, each number can only appear once inside this sub-grid. To make this sub-grid, I need to make r0 and c0 to make an imaginary box.

```
60  def print_board(bo):
61      for row in range(len(bo)):  # iterating rows
62          if row % 3 == 0 and row != 0:
63              # print seperator between rows every 3 row
64              print("- - - - - - - - - -")
65          for column in range(len(bo[0])):  # range(number of digits in a row)
66              if column % 3 == 0 and column != 0:
67                  print('|', end='')
68
69              if column == 8:  # on the last column, print without space at the end
70                  print(bo[row][column])
71              else:
72                  print(str(bo[row][column]) + ' ', end='')  # print the numbers
73
```

This is the function that prints the puzzle. Line 62 prints a separator every 3 rows. On line 65, it will print a bar every 3 column. This represents the sub-grid so that it is easier to see. And row 69-72 is just printing the number with space at the end except on column 8 or the last column.

```
44  def solver(bo):  # sudoku solver function
45      row, col = find_empty(bo)
46      if row is None:  # find returns None
47          return True  # there is no blank space, puzzle solved
48      else:
49          for n in range(1, 10):  # inserting number from 1-9 to the blank space
50              if insert_number(bo, row, col, n):
51                  bo[row][col] = n
52
53                  if solver(bo):
54                      return True  # recursion
55
56              bo[row][col] = 0  # backtracking
57          return False
```

This is the main function for this project. As the name implies, it solves the puzzle. Line 46 checks if there is still any blank space. If there is none, the puzzle is finished. Else, we will try to insert a number from 1-9 in an empty space. This number cannot break the board meaning it has to pass the insert_number function that has been explained before. If this number is valid, it will be inputted to the space. This will go through all the blank spaces by doing recursion. This means it will call the function again and find a new blank space for it to try a number. However, if no solution is possible, it will backtrack and go back to the previous blank space and set it to 0 as seen in line 56. It will use the for loop in line 49 again and try a new number that might fit the space. This is done until the board is filled.

IX. Project Link

https://github.com/1miaocat/Sudoku-solver

X. References

- http://www.counton.org/sudoku/rules-of-sudoku.php#:~:text=The%20classic%20Sudoku%20game%20involves,a%20row%2C%20column%20or%20box.