13. praktiskais darbs. 2. semestris

PU1. uzdevums 1.līmenis.

Sastādīt programmu, kas uzģenerē korekti aizpildītu Sudoku spēles lapiņu, ja ir zināms cik skaitļu ir atvērti.

1.līmenis - lapiņa sākotnēji uzģenerēta (aizpildīta) korekti ar visiem skaitļiem un tad izvēlas atklātos skaitļus un izdzēš pārējos.

2.līmenis - uzģenerē atvērtos skaitļus (visus uzreiz vai secīgi pa vienam) un pēc tam pārbauda, vai šis komplekts ir korekti atrisināms, bet ja nē, tad atkārto procesu, kamēr iegūst korekti aizpildāmu komplektu.

Kods:

```
# Programmas nosaukums: Sudoku spēlē ģenerācija (1.LĪMENIS)

# PU1. (1MPR13_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas uzģenerē korekti aizpildītu Sudoku spēles lapiņu, ja ir zināms cik skaitļu ir atvērti.

# 1.līmenis - lapiņa sākotnēji uzģenerēta (aizpildīta) korekti ar visiem skaitļiem un tad izvēlas atklātos skaitļus un izdzēš pārējos.

# 2.līmenis - uzģenerē atvērtos skaitļus (visus uzreiz vai secīgi pa vienam) un pēc tam pārbauda, vai šis komplekts ir korekti atrisināms, bet ja nē, tad atkārto procesu,

# kamēr iegūst korekti aizpildāmu komplektu.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



"""

1.LĪMENIS TIKA IZPILDĪTS AR TĀ SAUCĀMO "BACKTRACKING" ALGORITMU.

"""


import random
```

```
def check_array_rows_and_columns(a):
    # Pārbauda vai matricā katra rindā un kolonnā visi skaitļi ir dažādi, izmantojot kopas.
    # Atgriež True, ja visi skaitļi visas rindas un kolonnas ir dažādi.
    # Atgriež False, ja ir kādi divi skaitļi kāda rinda vai kolonna kuri sakrīt.
    # a - divdimensijas masīvs (matrica).

    for i in range(9):
        # Pārbauda, vai katrā rindā nav skaitļu dublikātu (nav vienādu skaitļu).
        # Izmantojam kopas. Ja kopā nav ar gārumu 9, tad ir cipari kas atkartojas.
        if len(set(a[i])) != 9:
            print(str(i + 1) + ". rindā ir cipari, kas atkārtojas.")  # Izvadam, kur tika atrāsta kļūda.
            return False

    for j in range(9):
        # Pārbauda, vai katrā kolonnā nav skaitļu dublikātu (nav vienādu skaitļu).
        # Izmantojam kopas. Ja kopā nav ar gārumu 9, tad ir cipari kas atkartojas.
        col_numbers = [a[i][j] for i in range(9)]
        if len(set(col_numbers)) != 9:
            print(str(j + 1) + ". kolonnā ir cipari, kas atkārtojas.")
            return False

    return True


def check_submatrix(matrix, i, j):
    # Atgriež True ja 3x3 apakšmatricā (apakšmatricas ir paradītas zemāk) visi skaitļi ir dažādi.
    # Atgriež False ja 3x3 apakšmatricā kādi divi skaitļi ir vienādi.
    # Sudoku 3x3 apakšmatricas.
    # matrix - pilnā 9x9 matrica (divdimensijas masīvs).
    # i - no kuras rīndas sāksim (int).
```

```
    # j - no kuras kolonnas sāksim (int).

    submatrix = []
    for row in range(i, i + 3):
        for col in range(j, j + 3):
            submatrix.append(matrix[row][col])
    return len(set(submatrix)) == 9


def check_3x3_submatrixes(a):
    # Pārbaudam katru 3x3 apakšmatricu un paziņojam lietotājam kāda apakšmatrica skaitļi ir dažādi un kurā ir vienādi.
    # Izsauc check_submatrix(a, i, j) un paziņo lietotājam "Visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi.",
    # vai "Ne visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi."
    # Atgriež True, ja nav nevienas apakšmatricas, kurai iekšā ir divi vienādi skaitļi.
    # Atgriež False, ja ir kaut viena apakšmatrica, kurai iekšā ir divi vienādi skaitļi.
    # a - divdimensiju masīvs.

    # Sudoku deviņas apakšmatricas:
    # [0][0] [0][1] [0][2]   [0][3] [0][4] [0][5]   [0][6] [0][7] [0][8]
    # [1][0] [1][1] [1][2]   [1][3] [1][4] [1][5]   [1][6] [1][7] [1][8]
    # [2][0] [2][1] [2][2]   [2][3] [2][4] [2][5]   [2][6] [2][7] [2][8]

    # [3][0] [3][1] [3][2]   [3][3] [3][4] [3][5]   [3][6] [3][7] [3][8]
    # [4][0] [4][1] [4][2]   [4][3] [4][4] [4][5]   [4][6] [4][7] [4][8]
    # [5][0] [5][1] [5][2]   [5][3] [5][4] [5][5]   [5][6] [5][7] [5][8]

    # [6][0] [6][1] [6][2]   [6][3] [6][4] [6][5]   [6][6] [6][7] [6][8]
    # [7][0] [7][1] [7][2]   [7][3] [7][4] [7][5]   [7][6] [7][7] [7][8]
    # [8][0] [8][1] [8][2]   [8][3] [8][4] [8][5]   [8][6] [8][7] [8][8]
```

```
    for i in range(0, 9, 3):
        for j in range(0, 9, 3):
            if check_submatrix(a, i, j):
                pass
                #print(f"Visi skaitļi apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir atšķirīgi.")
            else:
                #print(f"Kļūda! Apakšmatricā [{i//3 + 1}][{j//3 + 1}] ir divi vienādi skaitļi!")
                return False
    return True



def generate_sudoku():
    # Izveidojam 9x9 nulles matricu.
    matrix = []
    for i in range(9):
        row = [0] * 9
        matrix.append(row)

    # Aizpildam matricu, sākot no augšējā kreisā stūra.
    fill_matrix(matrix, 0, 0)

    return matrix



def fill_matrix(matrix, i, j):
    # Aizpildām nulles matricu ar (backtracking algorithm) algoritmu tā, lai visas rindas būtu atšķirīgi skaitli.

    if i == 9:
        return True
```

```
    # Aprēķina nākamo "šūnu" indeksus.
    next_i = i + (j + 1) // 9
    next_j = (j + 1) % 9

    # Izveidot sarakstu ar veseliem skaitļiem no 1 līdz 9.
    list_random = list(range(1, 10))
    # Sajaucam sarakstā vērtības.
    random.shuffle(list_random)

    # Mēģīnām ielikt pēc kārtas katru ciparu no nejauši sajaukta saraksta.
    for number in list_random:
        # Pārbaudam, vai skaitlis ir derīgs (ar karogu "valid").
        valid = True

        # Pārbaudam rindas.
        for k in range(9):
            if matrix[i][k] == number:
                valid = False  # Ja sakrīt kāds skaitlīs rindā ar izvelēto skaitli, tad karogs ir False, vajag paņemt citu skaitli no list_random (ņēm pēc kārtas).
                break

        # Pārbaudam kolonnas.
        for k in range(9):
            if matrix[k][j] == number:
                valid = False  # Ja sakrīt kāds skaitlīs kolonna ar izvelēto skaitli, tad karogs ir False, vajag paņemt citu skaitli no list_random (ņēm pēc kārtas).
                break

        # Pārbaudam 3x3 noteiktas apakšmatricas.
        sub_i = i // 3 * 3  # "apakš_i"
        sub_j = j // 3 * 3  # "apakš_j"
```

```
        for k in range(sub_i, sub_i + 3):
            for l in range(sub_j, sub_j + 3):
                if matrix[k][l] == number:
                    valid = False
                    break

        # Ja skaitlis ir derīgs, aizpildam to "šūnu" un rekursīvi sākam aizpildīt nākamo "šūnu".
        if valid:
            matrix[i][j] = number
            if fill_matrix(matrix, next_i, next_j):
                return True

    # Ja esam izmēģinājuši visus skaitļus un neviens nedēr, tad jāatkāpjas par vienu soli atpakaļ, un jau citus skaitļus likt. (backtracking algorithm)
    matrix[i][j] = 0
    return False


def delete_cells(matrix, delete_count):
    # Izveidojam matricu, kas piepildīta ar nullēm
    result = []
    for i in range(len(matrix)):
        row = [0] * len(matrix[0])
        result.append(row)

    # Aizpildam noteiktu skaitu "šūnu" ar skaitļiem no 1 līdz 9
    fill_count = len(matrix) * len(matrix[0]) - delete_count
    for k in range(fill_count):
        i = random.randint(0, 8)
        j = random.randint(0, 8)
        while result[i][j] != 0:
```

```
            i = random.randint(0, 8)
            j = random.randint(0, 8)
        result[i][j] = matrix[i][j]

    return result


def matrix_to_string_float_3x3(matrix):
    # Atgriež matricas virknes attēlojumu, kur katra rinda ir atdalīta ar \n un izlīdzināta atbilstoši maksimālās vērtības garumam.
    # Ja vērtība ir vesels skaitlis, tā tiek parādīta bez komata. Pretējā gadījumā tas tiek parādīts ar decimālzīmi.
    # Funkcija arī atrod maksimālo vērtību garumu matricā un aizpilda nepieciešamās atstarpes " ", lai tās glīti izlīdzinātu (glītas atkāpes).
    # Ja matricā ir 0, tas tiek aizstāts ar simbolu ·.
    # matrix - matrica (divdimensiju masīvs ar izmēriem n x m).

    # Piemērs, kāda veida tiek atgriezta simbolu virkne:
    # 1 6 3  9 3 4  3 6 6
    # 4 9 9  2 7 3  9 9 5
    # 3 5 9  5 2 7  9 7 4
    #
    # 4 6 6  3 3 8  2 5 3
    # 1 5 6  8 9 2  4 8 3
    # 9 3 9  6 8 7  2 8 2
    #
    # 7 4 9  3 9 3  7 1 1
    # 1 3 5  2 6 3  1 3 1
    # 6 5 3  8 9 7  7 1 8

    rindas = len(matrix)
    kolonnas = len(matrix[0])
```

```
    max_len = 0


    for i in range(rindas):  # atrod max_len, lai noteiktu nepieciešamo atkāpi.
        for j in range(kolonnas):
            number = matrix[i][j]
            if number == int(number):
                value_len = len(str(int(number)))
            else:
                value_len = len(str(float(number)))
            if value_len > max_len:
                max_len = value_len


    # Izveido matricas virknes attēlojumu, kur katra rinda tiek atdalīta ar \n un izlīdzināta
atbilstoši maksimālās vērtības garumam
    sv = ""
    for i in range(rindas):
        for j in range(kolonnas):
            number = matrix[i][j]
            if number == 0:
                number = "·"
            elif number == int(number):
                number = int(number)
            else:
                number = str(float(number))
            atkape = " " * (max_len - len(str(number)))
            sv += atkape + str(number)
            if j < kolonnas - 1 and (j + 1) % 3 != 0:
                sv = sv + " "
            elif j < kolonnas - 1 and (j + 1) % 3 == 0:
                sv = sv + "  "
        sv = sv + "\n"
```

```
        if (i + 1) % 3 == 0 and i < rindas - 1:
            sv = sv + "\n"

    return sv


def is_natural_or_zero(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nulle, vai nav
    # Ja ir naturāls skaitlis vai nulle, tad True. Ja nav tad False.
    # n - simbolu virkne, kuru pārbauda.
    if str(n).isdigit() and float(n) == int(n) and int(n) >= 0:
        return True
    else:
        return False

# ----------------------------------------------------------
# Galvenā programmas daļa
# ----------------------------------------------------------


b = input("Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> ")
while not is_natural_or_zero(b) or int(b) > 81:
    b = input("Kļūda! Ievadiet veselu skaitli no 0 līdz 81. Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> ")
b = int(b)

izd = 81 - b

a = generate_sudoku()

if check_array_rows_and_columns(a) and check_3x3_submatrixes(a):
```

```
    print("Korekti aizpildīta Sudoku spēlēs lapiņa:")
    print()
else:
    print("Nav korekti aizpildīta Sudoku spēlēs lapiņa:")


print(matrix_to_string_float_3x3(a))


print("-------------------\n")


c = delete_cells(a, izd)


print("Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:\n")
print(matrix_to_string_float_3x3(c))
```

Testa piemēri:

1)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 100
Kļūda! Ievadiet veselu skaitli no 0 līdz 81. Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> -1
Kļūda! Ievadiet veselu skaitli no 0 līdz 81. Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> labi
Kļūda! Ievadiet veselu skaitli no 0 līdz 81. Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 0
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

-------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

. . .  . . .  . . .
. . .  . . .  . . .
. . .  . . .  . . .

. . .  . . .  . . .
. . .  . . .  . . .
. . .  . . .  . . .

. . .  . . .  . . .
. . .  . . .  . . .
. . .  . . .  . . .
```

2)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 30
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

1 · ·  · · ·  · · ·
7 · 5  8 · 4  · 2 ·
· 3 ·  · · 5  · · ·

· 9 ·  · 8 ·  · · ·
· · 2  · 5 1  4 · 3
· 5 1  · · ·  · · ·

· · ·  3 2 8  9 · ·
8 · 3  · 6 9  · · 5
· 1 ·  5 4 ·  · · 8
```

3)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 81
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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

4)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 50.5
Kļūda! Ievadiet veselu skaitli no 0 līdz 81. Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 50
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

· · ·  1 · ·  9 · 2
7 9 1  · · ·  · 5 ·
2 4 3  5 9 6  1 · 7
```

5)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 15
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

· 9 ·  · · ·  · · ·
· 7 8  · · ·  2 · ·
· · ·  · · ·  · · ·

· · 4  6 · ·  · · 9
· 6 ·  · · ·  · · ·
1 · 7  · · ·  · · ·

· · ·  · · ·  · · 1
· · ·  · · 1  · · 7
3 · ·  · 7 ·  · · ·
```

6)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 66
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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

7)

```
Ievadiet cik skaitļu ir atvērti Sudoku lapiņai ==> 80
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

------------------

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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