13. praktiskais darbs. 2. semestris

1. uzdevums

Sastādīt programmu, kas pārbauda vai Sudoku spēles lapiņa ir aizpildīta korekti. (2 punkti)

Kods:

```
# Programmas nosaukums: Sudoku spēles lapiņa korektas aizpildīšanas pārbaude

# 1. uzdevums (1MPR13_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas pārbauda vai Sudoku spēles lapiņa ir aizpildīta korekti. (2 punkti)

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import numpy



def fill_array_randomly(n, m):
    # Nejauši aizpilda n x m matricu ar naturāliem skaitļiem no 1 līdz 9.
    # [1, 2, 3, 4, 5, 6, 7, 8, 9].
    # n - rindu skaits matricā (sudoku ir 9 rindas) (int).
    # m - kolonnu skaits matricā (sudoku ir 9 kolonnas) (int).
    # Atgriež aizpildīto masīvu ar nejaušiem naturāliem skaitļiem no 1 līdz 9.

    # Izveidojam tukšu 9x9 divdimensijas masīvu (matricu).
    a = numpy.zeros((n, m), dtype=int)

    # Ejam ciklā cauri katrai masīva rindai un kolonnai.
    for i in range(n):
```

```
        for j in range(m):
            # Ģenerējam nejaušu naturālu skaitli no 1 līdz 9.
            random_number = numpy.random.randint(1, 10)
            # Piešķiram [i][j] vietā nejaušu naturālu skaitli no 1 līdz 9.
            a[i][j] = random_number

    # Atgriež aizpildīto masīvu ar nejaušiem naturāliem skaitļiem no 1 līdz 9.
    return a



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):
                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 input_sudoku_matrix():
    # Paprasa lietotājam ievādīt veselu skaitļi no 1 līdz 9 katrai "šūnas" vērtībai 9x9 matricai.
    # Atgriež aizpildītu ar lietotāja ievādītam vērtībam matricu ar veseliem skaitļiem no 1 līdz
9.
    # Ejam ciklā caur katru masīva rindu un kolonnu.

    for i in range(9):
        for j in range(9):
            # Turpinam palūgt ievadi, līdz tiek norādīts derīgs skaitlis (cipars no 1 līdz 9).
            while True:
                # Palūdzam lietotājam ievadīt pašreizējās pozīcijas skaitļi (cipars no 1 līdz 9).
                number = input("Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [" + str(i) + "][" + str(j) +
"]: ")
```

```
            # Pārbaudam, vai ievadītais skaitlis ir no 1 līdz 9 un vai vispār tas ir cipars.
            if number.isdigit() and 1 <= int(number) <= 9:
                # Konvertējam ievadi par veselu skaitli un piešķiram to masīvam.
                a[i][j] = int(number)
                break
            else:
                print("Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.")

    # Agriež aizpildītu ar lietotāja ievādītam vērtībam matricu ar veseliem skaitļiem no 1 līdz 9.
    return a


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 numpy 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 += "\n"

    return sv



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



a = numpy.zeros((9, 9))

choose = input("Vai gribat nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> ")
while choose != "n" and choose != "m":
    choose = input("Kļūda! Ievadīet \"n\" vai \"m\"! Vai gribāt nejauši aizpildīt Sudoku kartīti vai manuāli? (n/m) ==> ")

if choose == "n":
    a = fill_array_randomly(9, 9)
    print()
    print(matrix_to_string_float(a))

elif choose == "m":
    b = input_sudoku_matrix()
    print()
```

```
    print(matrix_to_string_float(b))


if check_array_rows_and_columns(a) and check_3x3_submatrixes(a):
    print("Ir korekti aizpildīta Sudoku spēlēs lapiņa.")  # VALID
else:
    print("Nav korekti aizpildīta Sudoku spēlēs lapiņa.")  # INVALID



# TESTĒŠANAI
'''
for i in range(1000):
    a = fill_array_randomly(9, 9)
    # Izdrukām masīvu
    print()
    print(matrix_to_string_float(a))
    # if check_array_rows_and_columns(a) and check_3x3_submatrixes(a):
    if check_3x3_submatrixes(a):
        print("VALID")
    else:
        print("INVALID")
'''


# TESTĒŠANAI
'''
for i in range(1000):
    a = fill_array_randomly(9, 9)
    # Izdrukām masīvu
    print()
    print(matrix_to_string_float(a))
    if check_array_rows_and_columns(a):
        print("VALID")
```

```
    else:
        print("INVALID")
'''
```

Testa piemēri:

1)

```
Vai gribāt nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> n

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

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

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

1. rindā ir cipari, kas atkārtojas.
Nav korekti aizpildīta Sudoku spēlēs lapiņa.
```

2)

```
Vai gribāt nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> labi
Kļūda! Ievadiet "n" vai "m"! Vai gribāt nejauši aizpildīt Sudoku kartīti vai manuāli? (n/m) ==> labi
Kļūda! Ievadiet "n" vai "m"! Vai gribāt nejauši aizpildīt Sudoku kartīti vai manuāli? (n/m) ==> 5
Kļūda! Ievadiet "n" vai "m"! Vai gribāt nejauši aizpildīt Sudoku kartīti vai manuāli? (n/m) ==> 5.3
Kļūda! Ievadiet "n" vai "m"! Vai gribāt nejauši aizpildīt Sudoku kartīti vai manuāli? (n/m) ==> n

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

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

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

1. rindā ir cipari, kas atkārtojas.
Nav korekti aizpildīta Sudoku spēlēs lapiņa.
```

3)

Vai gribat nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> m

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][1]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][2]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][4]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][7]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][0]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][1]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][2]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][3]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][4]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][5]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][7]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]:

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 22

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 22

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][8]: 2

2 1 9 5 4 3 6 7 8

5 4 3 8 7 6 9 1 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2

3. rindā ir cipari, kas atkārtojas.

Nav korekti aizpildīta Sudoku spēlēs lapiņa.

```
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][2]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][3]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][4]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][5]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][6]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][7]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][8]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][0]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][1]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][2]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][3]: 22
Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][3]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][4]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][5]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][6]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][7]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][8]: 2

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

2 2 2  2 2 2  2 2 2
2 2 2  2 2 2  2 2 2
2 2 2  2 2 2  2 2 2

2 2 2  2 2 2  2 2 2
2 2 2  2 2 2  2 2 2
2 2 2  2 2 2  2 2 2

3. rindā ir cipari, kas atkārtojas.
Nav korekti aizpildīta Sudoku spēlēs lapiņa.
```

4)

Vai gribāt nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> m

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][1]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][2]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][4]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][7]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][0]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][1]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][2]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][3]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][4]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][5]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][7]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][0]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][1]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][2]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][4]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][5]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][6]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][7]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][8]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][0]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][1]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][3]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][4]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][5]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][6]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][7]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][8]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][0]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][1]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][2]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][3]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][4]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][5]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][7]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][8]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][0]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][1]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][4]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][5]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][6]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][7]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][0]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][1]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][4]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][5]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][6]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][8]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][0]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][1]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][2]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][3]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][4]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][5]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][7]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][8]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][0]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][1]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][2]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][7]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][8]: 7

2 1 9 5 4 3 6 7 8

5 4 3 8 7 6 9 1 2

8 7 6 2 1 9 3 4 5

4 3 2 7 6 5 8 9 1

7 6 5 1 9 8 2 3 4

1 9 8 2 3 4 1 9 8

4 3 2 5 6 7 3 2 1

6 5 4 7 8 9 6 5 4

9 8 7 1 2 3 9 8 7

6. rindā ir cipari, kas atkārtojas.

Nav korekti aizpildīta Sudoku spēlēs lapiņa.

```
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][0]: 6
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][1]: 5
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][2]: 4
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][3]: 7
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][4]: 8
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][5]: 9
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][6]: 6
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][7]: 5
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][8]: 4
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][0]: 9
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][1]: 8
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][2]: 7
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][3]: 1
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][4]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][5]: 3
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][6]: 9
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][7]: 8
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][8]: 7

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

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

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

6. rindā ir cipari, kas atkārtojas.
Nav korekti aizpildīta Sudoku spēlēs lapiņa.
```

5)

Vai gribat nejauši aizpildīt Sudoku spēlēs lapiņu vai manuāli? (n/m) ==> m

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 10

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 0

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 12.2

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 5.5

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: labi

Nepareiza ievade! Lūdzu, ievadiet veselu skaitli no 1 līdz 9.

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][0]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][1]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][2]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][3]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][4]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][5]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][6]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][7]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [0][8]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][0]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][1]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][2]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][3]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][4]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][5]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][6]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][7]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [1][8]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][0]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][1]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][2]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][3]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][4]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][5]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][6]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][7]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [2][8]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][0]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][1]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][2]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][3]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][4]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][5]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][6]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][7]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [3][8]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][0]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][1]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][2]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][3]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][4]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][5]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][6]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][7]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [4][8]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][0]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][1]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][2]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][3]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][4]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][5]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][6]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][7]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [5][8]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][0]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][1]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][2]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][3]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][4]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][5]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][6]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][7]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [6][8]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][0]: 6

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][1]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][2]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][3]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][4]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][5]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][6]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][7]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [7][8]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][0]: 9

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][1]: 8

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][2]: 7

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][3]: 3

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][4]: 2

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][5]: 1

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][6]: 4

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][7]: 5

Ievadiet veselu skaitli no 1 līdz 9 pozīcijai [8][8]: 6

2 1 9 5 4 3 6 7 8

5 4 3 8 7 6 9 1 2

8 7 6 2 1 9 3 4 5

4 3 2 7 6 5 8 9 1

7 6 5 1 9 8 2 3 4

1 9 8 4 3 2 5 6 7

3 2 1 6 5 4 7 8 9

6 5 4 9 8 7 1 2 3

9 8 7 3 2 1 4 5 6

Visi skaitļi apakšmatricā [1][1] ir atšķirīgi.

Visi skaitļi apakšmatricā [1][2] ir atšķirīgi.

Visi skaitļi apakšmatricā [1][3] ir atšķirīgi.

Visi skaitļi apakšmatricā [2][1] ir atšķirīgi.

Visi skaitļi apakšmatricā [2][2] ir atšķirīgi.

Visi skaitļi apakšmatricā [2][3] ir atšķirīgi.

Visi skaitļi apakšmatricā [3][1] ir atšķirīgi.

Visi skaitļi apakšmatricā [3][2] ir atšķirīgi.

Visi skaitļi apakšmatricā [3][3] ir atšķirīgi.

Ir korekti aizpildīta Sudoku spēlēs lapiņa.

```
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [7][8]: 3
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][0]: 9
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][1]: 8
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][2]: 7
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][3]: 3
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][4]: 2
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][5]: 1
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][6]: 4
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][7]: 5
Ievadiet veselu skaitli no 1 līdz 9 pozicijai [8][8]: 6

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

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

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

Visi skaitļi apakšmatricā [1][1] ir atšķirīgi.
Visi skaitļi apakšmatricā [1][2] ir atšķirīgi.
Visi skaitļi apakšmatricā [1][3] ir atšķirīgi.
Visi skaitļi apakšmatricā [2][1] ir atšķirīgi.
Visi skaitļi apakšmatricā [2][2] ir atšķirīgi.
Visi skaitļi apakšmatricā [2][3] ir atšķirīgi.
Visi skaitļi apakšmatricā [3][1] ir atšķirīgi.
Visi skaitļi apakšmatricā [3][2] ir atšķirīgi.
Visi skaitļi apakšmatricā [3][3] ir atšķirīgi.
Ir korekti aizpildīta Sudoku spēlēs lapiņa.
```

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 atpakāļ, un jau citus skaitļus likt. (backtracking algorithm)
    matrix[i][j] = 0
    return False


def delete_cells(matrix, delete_count):
    # Atgriež matricu, kur nejauši tiek izvelēti matricas skaitļi delete_count skaitā un tie tiek pārverti par nullem.
    # matrix - divdimensijas masīvs (matrica), kurai gribām pārverst delete_count skaitļus.
    # delete_count - cik skaitļus gribām "pārverst" par nullem. Jo lielāk, jo ir lielāka Sudoku spēlēs grūtība.

    for k in range(delete_count):
        i = random.randint(0, 8)
        j = random.randint(0, 8)
        matrix[i][j] = 0
    return matrix


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


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

```
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, 40)

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

print(matrix_to_string_float_3x3(c))
```

Testa piemēri:

1)

```
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

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

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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

2)

```
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

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

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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

3)

```
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

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

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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

4)

```
Korekti aizpildīta Sudoku spēlēs lapiņa:

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

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

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

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

Sudoku lapiņa ar nejauši izdzēstiem skaitļiem:

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

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

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

2. uzdevums

Sastādīt programmu, kas realizē pamatdarbības ar kompleksiem skaitļiem.

Kods:

```
# Programmas nosaukums: Kompleksa skaitļi

# 2.uzdevums (1MPR13_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē pamatdarbības ar kompleksiem skaitļiem.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



import math



class ComplexNumber:

    # Kompleksu skaitļu klase.

    def __init__(self, re=0, im=0):

        # Pēc noklusējuma izveido tukšu komplēksa skaitli (0 + 0i)

        # Ja ir norādots citādi, tad izveido tā, ka ievadīja lietotājs.

        self.re = re

        self.im = im


    def __repr__(self):

        # print()

        # Kompleksā skaitļu izvadīšanai lietotājam.

        if self.re != 0:  # Ja ir kāda reāla daļa, tad izvadam komplēkso skaitli ar realu daļu (neviss 0 + i*n)


            if self.im > 0 and self.im != 1:  # Ja imaginära daļa nav 1 un tā ir lielāka par 0, tad rakstām n + i, neviss n + k*i
```

```
            return f"{self.re} + {self.im}i"

        elif self.im == 1:  # Ja imaginära daļa ir 1, tad rakstām n + i, neviss n + 1*i
            return f"{self.re} + i"

        elif self.im == 0:  # Ja imaginära daļa ir 0, tad rakstām tikai reālu daļu n
            return f"{self.re}"

        elif self.im == -1:  # Ja imaginära daļa ir -1, tad rakstām n - i, neviss n - 1*i
            return f"{self.re} - i"

        else:  # Citā gadījumā rakstām n - k*i
            return f"{self.re} - {-self.im}i"

    else:  # Ja nav reālas daļas, tad nav jēgas rakstīt 0 + k*i, tad izvadam komplēkso skaitli tikai ar imagināru daļu k*i
        if self.im > 0 and self.im != 1:  # Ja imaginära daļa ir pozitīva un nav viens, tad izvadam k*i, neviss 0 + k*i
            return f"{self.im}i"

        elif self.im == 1:  # Ja imaginära daļa ir 1, tad izvadam i, neviss 0 + 1*i
            return "i"

        elif self.im == 0:  # Ja imaginära daļa ir 0, tad izvadam 0, neviss 0 + 0*i
            return "0"

        elif self.im == -1:  # Ja imaginära daļa ir 1, tad izvadam -i, neviss 0 - 1*i
            return "-i"

        else:  # Citādi izvādam -k*i (nav realas daļas un imaginära daļa ir negatīva un nav -1)
            return f"-{-self.im}i"
```

```
def arg(self):
    # Atgriež komplēksa skaitļa argumentu.
    return math.atan2(self.im, self.re)

def __add__(self, other):
    # +
    # Atgriež komplēksa skaitļa summu (self + other).
    real_sum = self.re + other.re
    imaginary_sum = self.im + other.im
    return ComplexNumber(real_sum, imaginary_sum)

def __iadd__(self, other):
    # +=
    # Atgriež komplēksa skaitļa summu (self + other), bet kā __iadd__ (+=).
    # Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.
    self.re += other.re
    self.im += other.im
    return self

def __sub__(self, other):
    # -
    # Atgriež komplēksa skaitļa starpību (self - other).
    real_diff = self.re - other.re
    imaginary_diff = self.im - other.im
    return ComplexNumber(real_diff, imaginary_diff)

def __isub__(self, other):
    # -=
    # Atgriež komplēksa skaitļa summu (self - other), bet kā __isub__ (-=).
    # Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.
    self.re -= other.re
```

```
    self.im -= other.im
    return self

def __mul__(self, other):
    # *
    # Atgriež komplēksa skaitļa reizinājumu (self * other).
    real_product = (self.re * other.re) - (self.im * other.im)
    imaginary_product = (self.re * other.im) + (self.im * other.re)
    return ComplexNumber(real_product, imaginary_product)

def __imul__(self, other):
    # *=
    # Atgriež komplēksa skaitļa reizinājumu (self * other) bet kā __imul__ (*=).
    # Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.
    re1 = self.re
    im1 = self.im
    self.re = (re1 * other.re) - (im1 * other.im)
    self.im = (re1 * other.im) + (im1 * other.re)
    return self

def __truediv__(self, other):
    # /
    # Atgriež komplēksa skaitļa dalījumu (self / other).
    denominator = (other.re * other.re) + (other.im * other.im)
    real_quotient = ((self.re * other.re) + (self.im * other.im)) / denominator
    imaginary_quotient = ((self.im * other.re) - (self.re * other.im)) / denominator
    return ComplexNumber(real_quotient, imaginary_quotient)

def __itruediv__(self, other):
    # /=
    # Atgriež komplēksa skaitļa dalījumu (self / other) bet kā __itruediv__ (/=).
```

```
    # Atgriež jau eksistējošu mainīgu, neviss izveido jaunu mainīgu.
    denominator = (other.re ** 2) + (other.im ** 2)
    real_quotient = ((self.re * other.re) + (self.im * other.im)) / denominator
    imaginary_quotient = ((self.im * other.re) - (self.re * other.im)) / denominator
    self.re = real_quotient
    self.im = imaginary_quotient
    return self

def __abs__(self):
    # Atgriež komplēksa skaitļa moduli.
    return math.sqrt(self.re * self.re + self.im * self.im)

def conjugate(self):
    # Atgriež komplēksa skaitļas komplēksa saistīto skaitli.
    return ComplexNumber(self.re, -self.im)

def __pow__(self, power):
    # Atgriež komplēksa skaitļi, kurš tika pacēlts naturāla pakāpe.
    modulus = self.__abs__() ** power
    arg = power * self.arg()
    re = modulus * math.cos(arg)
    im = modulus * math.sin(arg)
    return ComplexNumber(re, im)

def complex_power(z, n):
    # Atgriež komplēksa skaitļi, kurš tika pacēlts pakāpe.
    r = math.sqrt(z.re**2 + z.im**2)
    theta = math.atan2(z.im, z.re)
    re = r ** n * math.cos(n * theta)
    im_part = r ** n * math.sin(n * theta)
    return ComplexNumber(re, im_part)
```

```
    def n_roots(self, n):
        # Atgriež sarakstu, ar visiem komplēksa skaitļa saknēm.
        # n - kuru sakni gribām izvilkt
        roots = []
        modulus = abs(self)
        arg = self.arg()
        for k in range(n):
            root_argument = (arg + 2 * k * math.pi) / n
            re = modulus * math.cos(root_argument)
            im_part = modulus * math.sin(root_argument)
            roots.append(ComplexNumber(re, im_part))
        return roots

    def trigonometric_form(self):
        # Izvadīt lietotājam komplēksu skaitli trigonometriskajā formā.
        r = abs(self)
        theta = self.arg()
        return f"{r:.2f}(cos({theta:.2f}) + isin({theta:.2f}))"

    def exponent_form(self):
        # Izvadīt lietotājam komplēksu skaitli eksponenciāla formā.
        modulus = abs(self)
        arg = self.arg()
        return f"{modulus} * e^({arg}i)"


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

```
print("Operācijas ar komplēksa skaitliem:")
num1 = ComplexNumber(-1, -4)
num2 = ComplexNumber(1, 3)
# print(num1 + num4)
print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))


num1 = ComplexNumber(-1, 4)
num2 = ComplexNumber(1, -3)
# print(num1 + num4)
print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))


num1 = ComplexNumber(2, 4)
num2 = ComplexNumber(1, -3)
# print(num1 + num4)
print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))


num1 = ComplexNumber(-1, -4)
num2 = ComplexNumber(-1, 3)
# print(num1 + num4)
print("(" + str(num1) + ") + (" + str(num2) + ") = " + str(num1 + num2))


num3 = ComplexNumber(19, 15)
num4 = ComplexNumber(19, 15)
# print(num3 - num4)
print("(" + str(num3) + ") - (" + str(num4) + ") = " + str(num3 - num4))
```

```
# print(num1 / num2)
# print(num3 / num4)
print("(" + str(num1) + ") / (" + str(num2) + ") = " + str(num1 / num2))
print("(" + str(num3) + ") / (" + str(num4) + ") = " + str(num3 / num4))


# print(num1 * num2)
# print(num3 * num4)
print("(" + str(num1) + ") * (" + str(num2) + ") = " + str(num1 * num2))
print("(" + str(num3) + ") * (" + str(num4) + ") = " + str(num3 * num4))


# print(abs(num1))
# print(abs(num3))
print("|" + str(num1) + "| = " + str(abs(num1)))
print("|" + str(num3) + "| = " + str(abs(num3)))


# print(conjugate(num1))
# print(conjugate(num3))
print("conj(" + str(num1) + ") = " + str(ComplexNumber.conjugate(num1)))
print("conj(" + str(num3) + ") = " + str(ComplexNumber.conjugate(num3)) + "\n")


num5 = ComplexNumber(2, 4)
num6 = ComplexNumber(3, 6)
numiadd = ComplexNumber(3, 6)
numiadd1 = ComplexNumber(3, 6)

numiadd += num5
```

```
print("Izmantojot __iadd__:")
print("(" + str(numiadd1) + ") + (" + str(num5) + ") = " + str(numiadd) + "\n")

print("Izmantojot __add__:")
num_add = num5 + num6
print("(" + str(num5) + ") + (" + str(num6) + ") = " + str(num_add) + "\n")


numisub = ComplexNumber(3, 6)
numisub1 = ComplexNumber(3, 6)
numisub -= num5

print("Izmantojot __isub__:")
print("(" + str(numisub1) + ") - (" + str(num5) + ") = " + str(numisub) + "\n")


numisub = ComplexNumber(3, 6)
numisub1 = ComplexNumber(3, 6)
numisub = numisub1 - num5

print("Izmantojot __sub__:")
print("(" + str(numisub1) + ") - (" + str(num5) + ") = " + str(numisub) + "\n")


num7 = ComplexNumber(0.5, math.sqrt(3) / 2)
num8 = ComplexNumber(5, 5)
power1 = 20
print("Komplēksa skaitļa pakāpe:")
print("(" + str(num7) + ") ** " + str(power1) + " = " + str(num7 ** 20) + "\n")
```

```
z = ComplexNumber(3, 4)
z_cubed = ComplexNumber.complex_power(z, 3)
print("Komplēksa skaitļa pakāpe:")
print("(" + str(z) + ") ** " + str(3) + " = " + str(z_cubed) + "\n")


b = ComplexNumber(0.5, math.sqrt(3) / 2)
print("Komplēksa skaitļa saknes:")
print("sqrt(" + str(b) + ") = " + str(ComplexNumber.n_roots(b, 2)) + "\n")


b = ComplexNumber(0.5, math.sqrt(3) / 2)
print("Komplēksa skaitļa saknes:")
print("sqrt(" + str(b) + ") = " + str(ComplexNumber.n_roots(b, 4)) + "\n")


c = ComplexNumber(1, 1)
print("Komplēksa saistītajs:")
print("conj(" + str(c) + ") = " + str(ComplexNumber.conjugate(c)) + "\n")


d = ComplexNumber(-1, 1)
print("Pārveidot trigonometriskā formā:")
print("(" + str(d) + ") = " + str(ComplexNumber.trigonometric_form(d)) + "\n")


print("Pārveidot eksponenciālā formā:")
t = ComplexNumber(1, math.sqrt(3) / 3)
print("(" + str(t) + ") = " + str(ComplexNumber.exponent_form(t)) + "\n")
```

```
t = ComplexNumber(1, math.sqrt(3) / 3)
print("Komplēksa skaitļa arguments:")
print("arg(" + str(t) + ") = " + str(ComplexNumber.arg(t)) + "\n")


k = ComplexNumber(1, 2)
print("Komplēksa skaitļa izvadīšana (print):")
print(k)
print()


num5 = ComplexNumber(2, 4)
numimul = ComplexNumber(3, 6)
numimul1 = ComplexNumber(3, 6)

numimul *= num5
print("Izmantojot __imul__:")
print("(" + str(numimul1) + ") * (" + str(num5) + ") = " + str(numimul) + "\n")


num5 = ComplexNumber(2, 4)
numimul = ComplexNumber(3, 6)
numimul1 = ComplexNumber(3, 6)

numimul1 = numimul * num5
print("Izmantojot __mul__:")
print("(" + str(numimul) + ") * (" + str(num5) + ") = " + str(numimul1) + "\n")


num5 = ComplexNumber(2, 4)
numidiv = ComplexNumber(3, 6)
```

```
numidiv1 = ComplexNumber(3, 6)


numidiv /= num5
print("Izmantojot __itruediv__:")
print("(" + str(numidiv1) + ") / (" + str(num5) + ") = " + str(numidiv) + "\n")



num5 = ComplexNumber(2, 4)
numidiv = ComplexNumber(3, 6)
numidiv1 = ComplexNumber(3, 6)


numidiv1 = numidiv / num5
print("Izmantojot __truediv__:")
print("(" + str(numimul) + ") / (" + str(num5) + ") = " + str(numidiv1) + "\n")



print("Jaunais komplēksa skaitlis:")
new_num = ComplexNumber()
print(new_num)
```

Testa piemēri:

1)

Operācijas ar komplēksa skaitliem:

(-1 - 4i) + (1 + 3i) = -i

(-1 + 4i) + (1 - 3i) = i

(2 + 4i) + (1 - 3i) = 3 + i

(-1 - 4i) + (-1 + 3i) = -2 - i

(19 + 15i) - (19 + 15i) = 0

(-1 - 4i) / (-1 + 3i) = -1.1 + 0.7i

(19 + 15i) / (19 + 15i) = 1.0

(-1 - 4i) * (-1 + 3i) = 13 + i

(19 + 15i) * (19 + 15i) = 136 + 570i

|-1 - 4i| = 4.123105625617661

|19 + 15i| = 24.20743687382041

conj(-1 - 4i) = -1 + 4i

conj(19 + 15i) = 19 - 15i

Izmantojot __iadd__:

(3 + 6i) + (2 + 4i) = 5 + 10i

Izmantojot __add__:

(2 + 4i) + (3 + 6i) = 5 + 10i

Izmantojot __isub__:

(3 + 6i) - (2 + 4i) = 1 + 2i

Izmantojot __sub__:

(3 + 6i) - (2 + 4i) = 1 + 2i

Komplēksa skaitļa pakāpe:

(0.5 + 0.8660254037844386i) ** 20 = -0.4999999999999961 + 0.8660254037844384i

Komplēksa skaitļa pakāpe:

(3 + 4i) ** 3 = -117.0 + 44.000000000000036i

Komplēksa skaitļa saknes:

sqrt(0.5 + 0.8660254037844386i) = [0.8660254037844386 + 0.4999999999999999i, -0.8660254037844387 - 0.49999999999999967i]

Komplēksa skaitļa saknes:

sqrt(0.5 + 0.8660254037844386i) = [0.9659258262890682 + 0.2588190451025207i, -0.2588190451025206 + 0.9659258262890682i, -0.9659258262890682 - 0.25881904510252074i, 0.25881904510252024 - 0.9659258262890683i]

Komplēksa saistītajs:

conj(1 + i) = 1 - i

Pārveidot trigonometriskā formā:

(-1 + i) = 1.41(cos(2.36) + isin(2.36))

Pārveidot eksponenciālā formā:

(1 + 0.5773502691896257i) = 1.1547005383792515 * e^(0.5235987755982988i)

Komplēksa skaitļa arguments:

arg(1 + 0.5773502691896257i) = 0.5235987755982988

Komplēksa skaitļa izvadīšana (print):

1 + 2i

Izmantojot __imul__:

(3 + 6i) * (2 + 4i) = -18 + 24i

Izmantojot __mul__:

(3 + 6i) * (2 + 4i) = -18 + 24i

Izmantojot __itruediv__:

(3 + 6i) / (2 + 4i) = 1.5

Izmantojot __truediv__:

(3 + 6i) / (2 + 4i) = 1.5

Jaunais komplēksa skaitlis:

0

2)

```
Operācijas ar komplēksa skaitliem:
(-1 - 4i) + (1 + 3i) = -i
(-1 + 4i) + (1 - 3i) = i
(2 + 4i) + (1 - 3i) = 3 + i
(-1 - 4i) + (-1 + 3i) = -2 - i
(19 + 15i) - (19 + 15i) = 0
(-1 - 4i) / (-1 + 3i) = -1.1 + 0.7i
(19 + 15i) / (19 + 15i) = 1.0
(-1 - 4i) * (-1 + 3i) = 13 + i
(19 + 15i) * (19 + 15i) = 136 + 570i
|-1 - 4i| = 4.123105625617661
|19 + 15i| = 24.20743687382041
conj(-1 - 4i) = -1 + 4i
conj(19 + 15i) = 19 - 15i

Izmantojot __iadd__:
(3 + 6i) + (2 + 4i) = 5 + 10i

Izmantojot __add__:
(2 + 4i) + (3 + 6i) = 5 + 10i

Izmantojot __isub__:
(3 + 6i) - (2 + 4i) = 1 + 2i

Izmantojot __sub__:
(3 + 6i) - (2 + 4i) = 1 + 2i

Komplēksa skaitļa pakāpe:
(0.5 + 0.86602540378443864i) ** 20 = -0.4999999999999961 + 0.8660254037844384i

Komplēksa skaitļa pakāpe:
(3 + 4i) ** 3 = -117.0 + 44.000000000000036i
```

3)

```
Komplēksa saistītajs:
conj(1 + i) = 1 - i

Pārveidot trigonometriskā formā:
(-1 + i) = 1.41(cos(2.36) + isin(2.36))

Pārveidot eksponenciālā formā:
(1 + 0.57735026918962573i) = 1.1547005383792515 * e^(0.52359877559829883i)

Komplēksa skaitļa arguments:
arg(1 + 0.57735026918962573i) = 0.5235987755982988

Komplēksa skaitļa izvadīšana (print):
1 + 2i

Izmantojot __imul__:
(3 + 6i) * (2 + 4i) = -18 + 24i

Izmantojot __mul__:
(3 + 6i) * (2 + 4i) = -18 + 24i

Izmantojot __itruediv__:
(3 + 6i) / (2 + 4i) = 1.5

Izmantojot __truediv__:
(3 + 6i) / (2 + 4i) = 1.5

Jaunais komplēksa skaitlis:
0
```

3. uzdevums (I variants)

Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras (ar ierakstiem).

Kods:

```
# Programmas nosaukums: N-stūra laukums ar ierakstiem

# 3.uzdevums (1MPR13_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



"""

UZDEVUMS TIKA REALIZĒTS AR IERAKSTIEM.

"""



import types

import numpy



def input_polygon_coords(n):

    # Paprasa lietotājam ievadīt virsotnes X un Y koordinātas.

    # Atgriež ierakstu a, ar lietotāja ievadītam X un Y koordinātam.

    # n - naturāls skaitlis - daudzstūra virsotņu skaits.



    for i in range(1, n + 1):

        punkts = types.SimpleNamespace()



        temp_x = input("Ievadiet " + str(i) + ". virsotnes X koordināti ==> ")

        while not is_real_check(temp_x):
```

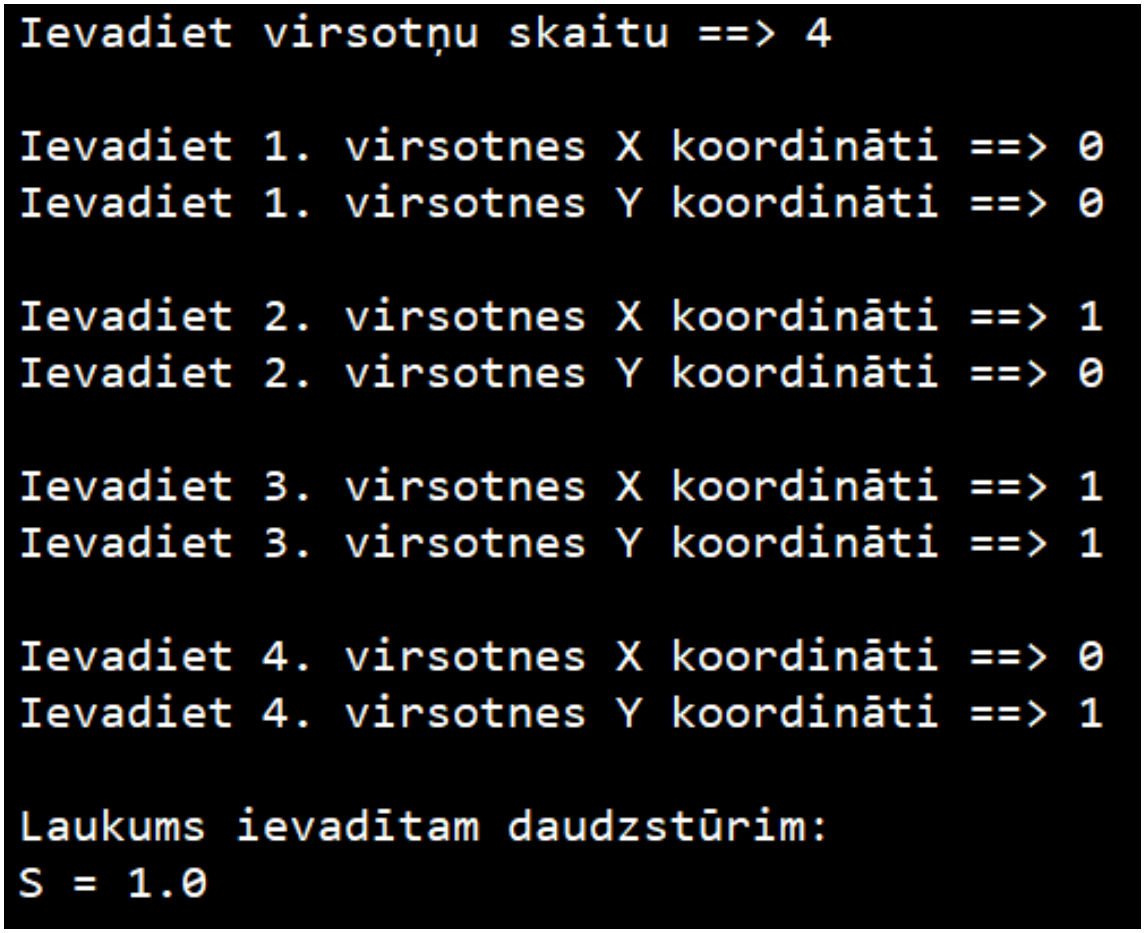
```
        temp_x = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i) + ". virsotnes X koordināti ==> ")

    temp_x = float(temp_x)

    temp_y = input("Ievadiet " + str(i) + ". virsotnes Y koordināti ==> ")
    while not is_real_check(temp_y):
        temp_y = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i) + ". virsotnes X koordināti ==> ")

    temp_y = float(temp_y)

    print()

    punkts.x = temp_x
    punkts.y = temp_y

    a[i] = punkts

  return a



def area_using_shoelace_methode_list(a):
  # Aprēķina laukumu daudzstūrim, kurā punktu koordinātas ir definētas ierakstā a.
  # a - ieraksts, kurā glābajas daudzstūra koordinātas.
  # Ieraksta piemērs: [0 namespace(x=1.0, y=2.0) namespace(x=3.0, y=4.0)].
  # Atgriež daudzstūra laukumu izmantojot "Shoelace formula".

  a[0] = a[n]  # Fiktīvais punkts.

  s = 0
```

```
    for i in range(n):
        x = a[i].x + a[i + 1].x
        y = a[i].y - a[i + 1].y
        s = s + x * y

    s = abs(s) / 2

    return s


def is_real_check(n):
    # Pārbauda vai simbolu virkne ir reāls skaitlis vai nav.
    # Atgriež True, ja tas ir reāls skaitlis (float).
    # Atgriež False, ja tas nav reāls skaitlis (float).
    # n - pārbaudāma simbolu virkne.

    try:
        float(n)
    except:
        return False
    else:
        return True


def is_natural(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, 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
```

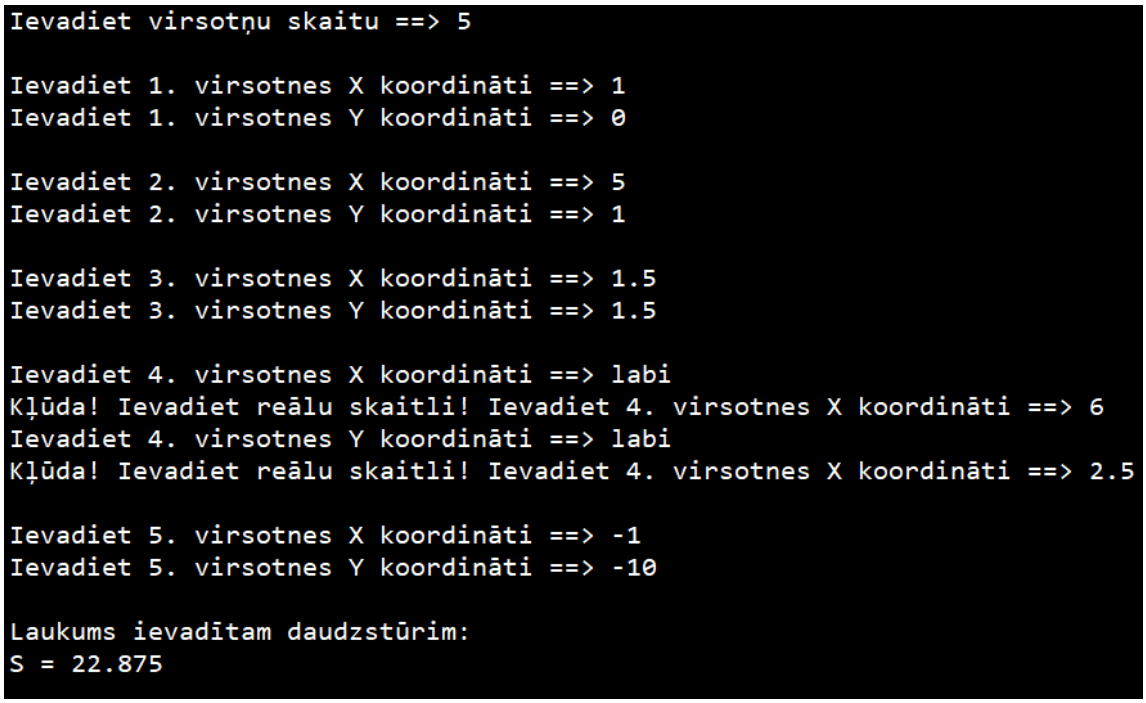
```
    else:
        return False



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



n = input("Ievadiet virsotņu skaitu ==> ")
while not is_natural(n):
    n = input("Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==>
")
n = int(n)

a = numpy.zeros(n + 1, "O")

print()
koord_saraksts = input_polygon_coords(n)
area = area_using_shoelace_methode_list(koord_saraksts)

print("Laukums ievadītam daudzstūrim:")
print("S = " + str(area))
```

Testa piemēri:

1)



2)



3)

```
Ievadiet virsotņu skaitu ==> 0
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> -1
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> 12.3
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> 2

Ievadiet 1. virsotnes X koordināti ==> 1
Ievadiet 1. virsotnes Y koordināti ==> 2

Ievadiet 2. virsotnes X koordināti ==> 3
Ievadiet 2. virsotnes Y koordināti ==> 4

Laukums ievadītam daudzstūrim:
S = 0.0
```

4)

```
Ievadiet virsotņu skaitu ==> 4

Ievadiet 1. virsotnes X koordināti ==> 1
Ievadiet 1. virsotnes Y koordināti ==> 1

Ievadiet 2. virsotnes X koordināti ==> 2
Ievadiet 2. virsotnes Y koordināti ==> 2

Ievadiet 3. virsotnes X koordināti ==> 3
Ievadiet 3. virsotnes Y koordināti ==> 3

Ievadiet 4. virsotnes X koordināti ==> -4
Ievadiet 4. virsotnes Y koordināti ==> 4

Laukums ievadītam daudzstūrim:
S = 8.0
```

3. uzdevums (II variants)

Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras. (ar vārdnīcām).

Kods:

```
# Programmas nosaukums: N-stūra laukums ar vārdnīcām

# 3.uzdevums (1MPR13_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas aprēķina N-stūra laukumu, tā virsotņu koordinātas ievadot no tastatūras.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0


"""

UZDEVUMS TIKA REALIZĒTS AR VĀRDNĪCAM.

"""


import types



def area_using_shoelace_methode_dict(vardnica, n):

    # Aprēķina laukumu daudzstūrim, kurā punktu koordinātas ir definētas vārdnīcā, šādā veidā {"Koordinātas nosaukums" : (x, y)}.

    # Atgriež daudzstūra laukumu izmantojot "Shoelace formula".

    # vardnica - vārdnīca, kurā ir saraksts ar koordinātam šadā veidā {"Koordinātas nosaukums" : (x, y)}.

    # n - naturāls skaitlis - daudzstūra virsotņu skaits.

    # Atgriež daudzstūra laukumu.


    s = 0


    for i in range(0, n):

        # Izmantojot Gausa formulu atrodam laukumu daudzstūrim (Shoelace formula).
```

```
        x = vardnica[t[i]].x + vardnica[t[i - 1]].x
        y = vardnica[t[i]].y - vardnica[t[i - 1]].y
        s = s + x * y

    s = abs(s) / 2

    return s


def create_dict_with_coords(n):
    # Paprasa lietotājam ievadīt virsotnes nosaukumus un tā X un Y koordinātas.
    # Izveido vārdnīcu tāda veidā {"Virsotnes nosaukums" : (x, y)}, kur katru "Virsotnes nosaukums" ievada lietotājs, (x, y) arī ievada lietotājs.
    # Atgriež vārdnīcu tāda veidā  {"Virsotnes nosaukums" : (x, y)}.
    # n - naturāls skaitlis - daudzstūra virsotņu skaits.

    i = 0

    for i in range(i, n):
        name = input("Ievadiet " + str(i + 1) + ". virsotnes nosaukumu ==> ")
        punkts = types.SimpleNamespace()

        temp_x = input("Ievadiet " + str(i + 1) + ". virsotnes X koordināti ==> ")
        while not is_real_check(temp_x):
            temp_x = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i + 1) + ". virsotnes X koordināti ==> ")

        temp_x = float(temp_x)

        temp_y = input("Ievadiet " + str(i + 1) + ". virsotnes Y koordināti ==> ")
        while not is_real_check(temp_y):
```

```
            temp_y = input("Kļūda! Ievadiet reālu skaitli! Ievadiet " + str(i + 1) + ". virsotnes X koordināti ==> ")

            temp_y = float(temp_y)

            punkts.x = temp_x
            punkts.y = temp_y

            vardnica.update({name: punkts})

            t.append(name)

    t.append("Fiktīvais punkts")
    vardnica["Fiktīvais punkts"] = vardnica[t[n - 1]]

    return vardnica


def is_natural(n):
    # Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav
    # Ja ir naturāls skaitlis, 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


def is_real_check(n):
    # Pārbauda vai simbolu virkne ir reāls skaitlis vai nav.
```

```
    # Atgriež True, ja tas ir reāls skaitlis (float).
    # Atgriež False, ja tas nav reāls skaitlis (float).
    # n - pārbaudāma simbolu virkne.

    try:
        float(n)
    except:
        return False
    else:
        return True



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



vardnica = {}
t = []

n = input("Ievadiet virsotņu skaitu ==> ")
while not is_natural(n):
    n = input("Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==>
")
n = int(n)

create_dict_with_coords(n)
s = area_using_shoelace_methode_dict(vardnica, n)

vardnica.pop("Fiktīvais punkts")  # izņemt no vārdnīcas atslēgu-vērtību pāri "Fiktīvais
punkts".
```

```
print()
print("Punktu sarakstu izvadīšanas veids:")
print("\"Punkta nosaukums\" : (x, y)")
print(vardnica)
print("S = " + str(s))
```

Testa piemēri:

1)

```
Ievadiet virsotņu skaitu ==> 4
Ievadiet 1. virsotnes nosaukumu ==> Dīķis
Ievadiet 1. virsotnes X koordināti ==> 0
Ievadiet 1. virsotnes Y koordināti ==> 0
Ievadiet 2. virsotnes nosaukumu ==> Māja
Ievadiet 2. virsotnes X koordināti ==> 1
Ievadiet 2. virsotnes Y koordināti ==> 0
Ievadiet 3. virsotnes nosaukumu ==> Lauks
Ievadiet 3. virsotnes X koordināti ==> 1
Ievadiet 3. virsotnes Y koordināti ==> 1
Ievadiet 4. virsotnes nosaukumu ==> Veikals
Ievadiet 4. virsotnes X koordināti ==> 0
Ievadiet 4. virsotnes Y koordināti ==> 1

Punktu sarakstu izvadīšanas veids:
"Punkta nosaukums" : (x, y)
{'Dīķis': namespace(x=0.0, y=0.0), 'Māja': namespace(x=1.0, y=0.0), 'Lauks': namespace(x=1.0, y=1.0), 'Veikals': namespace(x=0.0, y=1.0)}
S = 1.0
```

2)

```
Ievadiet virsotņu skaitu ==> 5
Ievadiet 1. virsotnes nosaukumu ==> Māja
Ievadiet 1. virsotnes X koordināti ==> 1
Ievadiet 1. virsotnes Y koordināti ==> 0
Ievadiet 2. virsotnes nosaukumu ==> Klēts
Ievadiet 2. virsotnes X koordināti ==> 5
Ievadiet 2. virsotnes Y koordināti ==> 1
Ievadiet 3. virsotnes nosaukumu ==> Maxima
Ievadiet 3. virsotnes X koordināti ==> 1.5
Ievadiet 3. virsotnes Y koordināti ==> 1.5
Ievadiet 4. virsotnes nosaukumu ==> Kaimiņi
Ievadiet 4. virsotnes X koordināti ==> Labi
Kļūda! Ievadiet reālu skaitli! Ievadiet 4. virsotnes X koordināti ==> OK
Kļūda! Ievadiet reālu skaitli! Ievadiet 4. virsotnes X koordināti ==> -6
Ievadiet 4. virsotnes Y koordināti ==> 2
Ievadiet 5. virsotnes nosaukumu ==> Drauga māja
Ievadiet 5. virsotnes X koordināti ==> -1
Ievadiet 5. virsotnes Y koordināti ==> -10

Punktu sarakstu izvadīšanas veids:
"Punkta nosaukums" : (x, y)
{'Māja': namespace(x=1.0, y=0.0), 'Klēts': namespace(x=5.0, y=1.0), 'Maxima': namespace(x=1.5, y=1.5), 'Kaimiņi': namespace(x=-6.0, y=2.0), 'Drauga māja': namespace(x=-1.0, y=-10.0)}
S = 45.5
```

3)

```
Ievadiet virsotņu skaitu ==> 3
Ievadiet 1. virsotnes nosaukumu ==> Nulles punkts
Ievadiet 1. virsotnes X koordināti ==> 0
Ievadiet 1. virsotnes Y koordināti ==> 0
Ievadiet 2. virsotnes nosaukumu ==> 1.punkts
Ievadiet 2. virsotnes X koordināti ==> -1
Ievadiet 2. virsotnes Y koordināti ==> -1
Ievadiet 3. virsotnes nosaukumu ==> 2.punkts
Ievadiet 3. virsotnes X koordināti ==> 2
Ievadiet 3. virsotnes Y koordināti ==> -1

Punktu sarakstu izvadīšanas veids:
"Punkta nosaukums" : (x, y)
{'Nulles punkts': namespace(x=0.0, y=0.0), '1.punkts': namespace(x=-1.0, y=-1.0), '2.punkts': namespace(x=2.0, y=-1.0)}
S = 1.5
```

4)

```
Ievadiet virsotņu skaitu ==> 6.5
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> -6
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> 0
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> pieci
Kļūda! Virsotņu skaitam ir jābūt naturālam skaitlim! Ievadiet virsotņu skaitu ==> 1
Ievadiet 1. virsotnes nosaukumu ==> Viens
Ievadiet 1. virsotnes X koordināti ==> 1
Ievadiet 1. virsotnes Y koordināti ==> 1

Punktu sarakstu izvadīšanas veids:
"Punkta nosaukums" : (x, y)
{'Viens': namespace(x=1.0, y=1.0)}
S = 0.0
```

4. uzdevums (Tekstuāla saskarne)

Sastādīt programmu, kas realizē iepirkšanos veikalā. Tiek nodrošināta iegādāto preču ievade (nosaukums, daudzums un cena) un "pirkumu čeka" izdrukāšana. (Vienkārša tekstuāla saskarne - 1 punkts, komplicēta grafiskā saskarne - 3 punkti.)

Kods (Tekstuāla saskarne):

```
# Programmas nosaukums: Veikals

# 4.uzdevums (1MPR13_Vladislavs_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē iepirkšanos veikalā. Tiek nodrošināta iegādāto preču ievade (nosaukums, daudzums un cena)

# un "pirkumu čeka" izdrukāšana. (Vienkārša tekstuāla saskarne - 1 punkts, komplicēta grafiskā saskarne - 3 punkti.)

# Programmas autors: Vladislavs Babaņins

# Versija 1.0



"""

UZDEVUMS TIKA REALIZĒTS AR TEKSTUALU SASKARNI.

"""



from random import randint



class Veikals:

    # Klase veikals.

    def __init__(self):

        # Izveidojam preces un selected_preces (izvelētas preces) vārdnīcas.

        # self.preces - vārdnica ar visam precem veikalā.

        # self.selected_preces - vārdnica ar lietotāja izvelētam precem.


        self.preces = dict()
```

```
        self.selected_preces = dict()


    def pievienot_preces(self, prece, cena, daudzums):
        # Metode veikala īpašniekam.
        # Preces pievienošana veikalām. Atjaunojam vārdnicu self.preces, preci, ar norādītu cenu un daudzumu.
        # prece - preces nosaukums (str).
        # cena - preces cena par vienu vienumu (float).
        # daudzums - cik daudz preci jāpiegāda veikalām (naturāls skaitlis lielāks par 0) (int).

        self.preces[prece] = {"cena": cena, "daudzums": daudzums}


    def dzest_preces(self, prece):
        # Metode veikala īpašniekam.
        # Nodzēs norādītu preci no veikalā, nodzēs noradītu preci no saraksta.
        # prece - preces nosaukums (str).

        print(f"Prece \"{prece}\" tika izmesta Getliņu izgāztuvē.")
        del self.preces[prece]


    def paradit_izveletas_preces(self):
        # Izdruka visas lietotaja izvelētas preces.

        print("Jūsu izvelētas preces:")
        for prece, info in self.selected_preces.items():
            cena = info["cena"]
            daudzums = info["daudzums"]
            print(f"{prece} - cena: {cena:.2f}€ , daudzums: {daudzums}")


    def paradit_preces(self):
        # Izdruka visas veikalā pieejamas preces lietotājam.
```

```
        print("Pieejamas preces veikalā:")
        for prece, info in self.preces.items():
            cena = info["cena"]
            daudzums = info["daudzums"]
            print(f"{prece} - cena: {cena:.2f}€ , daudzums: {daudzums}")

    def izveleties_preces(self, prece, daudzums):
        # Metode preces izvelēšanai lietotājam, jāizvēlas preci un to daudzumu.
        # prece - preces nosaukums (str).
        # daudzums - preces daudzums (int).

        if prece in self.preces:  # Ja tāda prece ir pieejama veikalā, tad tālak pārbaudam.
            if daudzums > self.preces[prece]["daudzums"]:  # Ja pieprasītais daudzums ir lielāks nekā ir veikalā, tad paziņojam, ka nevar nopirkt.
                print(f"Kļūda! Veikalā nav tik daudz preču! Precei \"{prece}\" pieejamais daudzums: {self.preces[prece]['daudzums']}")
            else:
                self.preces[prece]["daudzums"] -= daudzums  # Atņēmam nopirktu daudzumu.
                if prece not in self.selected_preces:
                    self.selected_preces[prece] = {"cena": self.preces[prece]["cena"], "daudzums": daudzums}
                else:
                    self.selected_preces[prece]["daudzums"] += daudzums  # Pievienojam daudzumu, cik daudz preces izvelējamies nopirkt.
                print(f"Izvēlētas preces: {prece}, cena: " + str(format(self.preces[prece]["cena"], ".2f")) + f"€ , daudzums: {daudzums}")
        else:
            print(f"Kļūda! Prece \"{prece}\" nav pieejama veikalā!")  # Ja tādas preces nav veikalā, tad paziņojam to.

    def atteikt_prece(self, prece, daudzums=None):
        # Lietotājs var atgriezt veikālā kādu no jau izvelētam precem ar norādīto daudzumu.
```

```
        # Ja lietotājs nenorāda daudzumu, tad tiek atgriezti visas preces.
        # prece - preces nosaukums (str).
        # daudzums - preces daudzums (int).

        if prece in self.selected_preces:
            if daudzums is None:  # Ja lietotājs nenorāda daudzumu, tad tiek atgriezti visas preces.
                daudzums = self.selected_preces[prece]["daudzums"]

            if daudzums > self.selected_preces[prece]["daudzums"]:  # Ja lietotājs grib atgriezt vairāk nekā paņem, tad kļūda.
                print(f"Kļūda! Precei \"{prece}\" ir izvēlēts mazāks daudzums. Nevar atgriezt vairāk nekā paņemat!")
            elif daudzums < 0:  # Ja lietotājs grib atgriezt negatīvu daudzumu, tad kļūda.
                print(f"Kļūda! Nevar atgriezt negatīvu daudzumu!")
            elif daudzums == 0:  # Ja lietotājs grib atgriezt neko, tad kļūda.
                print(f"Kļūda! Nevar atgriezt neko!")
            else:
                self.selected_preces[prece]["daudzums"] -= daudzums  # Atņēmam noradītu daudzumu no izvelētam precēm.
                self.preces[prece]["daudzums"] += daudzums  # Pievienojam noradītu daudzumu par izvelētam precēm.
                if self.selected_preces[prece]["daudzums"] == 0:  # Ja atņēmam tik daudz, ka vairāk tadas preces nav (nav izvelēta), tad nodzēsam to.
                    del self.selected_preces[prece]
                print(f"Izvēlētā prece \"{prece}\" tika atgriezta veikalā. Atgrieztais daudzums: {daudzums}")  # Paziņojam lietotājam informāciju.
        else:
            print(f"Kļūda! Prece netika \"{prece}\" izvēlēta.")  # Paziņojam lietotājam.

    def pirkumu_cekis(self):
        # Izdruka pirkumu lietotājam "pirkumu čeku", ar izvelētam precēm.
        total_price = 0
```

```
        print("---------- ČEKS ----------")
        print("SIA \"MAKSIMA\" Latvija")
        print("Veikals \"Maksima\"")
        print("Dārza iela 21, Bauska, t.22813371")
        print("Juridiskā adrese: \"Abras\", Krustkalni,")
        print("Ķekavas pagasts, Ķekavas novads, LV-2222")
        print("Čeks " + str(randint(100, 999)) + "/" + str(randint(100, 999)) + "                #" +
str(randint(10000000, 99999999)))
        print("========================================")

        for prece, info in self.selected_preces.items():
            cena = info["cena"]
            daudzums = info["daudzums"]
            preces_cena = cena * daudzums
            total_price += preces_cena
            print(f"{prece} - cena: {cena:.2f}€, daudzums: {daudzums}, {preces_cena:.2f}€")

        print("========================================")
        print(f"Kopā apmaksai: {total_price:.2f}€")
        print(f"Nopelnīta \"MAKSIMA\" nauda: {total_price * 0.01:.2f}€")
        print("----------------------------------------")



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



veikals = Veikals()


veikals.pievienot_preces("Āboli", 1.00, 50)
```

```
veikals.pievienot_preces("Piens", 1.40, 20)
veikals.pievienot_preces("Olas", 1.50, 30)
veikals.pievienot_preces("Pipari", 1.00, 100)

veikals.paradit_preces()
print()

veikals.dzest_preces("Piens")
print()

veikals.paradit_preces()
print()

veikals.izveleties_preces("Olas", 60)
veikals.izveleties_preces("Olas", 30)
veikals.izveleties_preces("Vārdnīca", 1)
veikals.izveleties_preces("Āboli", 50)
veikals.izveleties_preces("Pipari", 66)
print()

veikals.paradit_izveletas_preces()
print()

veikals.atteikt_prece("Āboli", 30)
veikals.atteikt_prece("Olas")
print()

veikals.paradit_izveletas_preces()
print()

veikals.atteikt_prece("Āboli", -10)
```

```
print()

veikals.paradit_izveletas_preces()
print()

veikals.atteikt_prece("Āboli", 0)
veikals.paradit_izveletas_preces()
print()
print()

veikals.pirkumu_cekis()
print()
print()

veikals.paradit_preces()
```

Testa piemēri:

1)

```
Pieejamas preces veikalā:
Āboli - cena: 1.00€ , daudzums: 50
Piens - cena: 1.40€ , daudzums: 20
Olas - cena: 1.50€ , daudzums: 30
Pipari - cena: 1.00€ , daudzums: 100

Prece "Piens" tika izmesta Getliņu izgāztuvē.

Pieejamas preces veikalā:
Āboli - cena: 1.00€ , daudzums: 50
Olas - cena: 1.50€ , daudzums: 30
Pipari - cena: 1.00€ , daudzums: 100
```

Kļūda! Veikalā nav tik daudz preču! Precei "Olas" pieejamais daudzums: 30

Izvēlētas preces: Olas, cena: 1.50€ , daudzums: 30

Kļūda! Prece "Vārdnīca" nav pieejama veikalā!

Izvēlētas preces: Āboli, cena: 1.00€ , daudzums: 50

Izvēlētas preces: Pipari, cena: 1.00€ , daudzums: 66

Jūsu izvelētas preces:

Olas - cena: 1.50€ , daudzums: 30

Āboli - cena: 1.00€ , daudzums: 50

Pipari - cena: 1.00€ , daudzums: 66

Izvēlētā prece "Āboli" tika atgriezta veikalā. Atgrieztais daudzums: 30

Izvēlētā prece "Olas" tika atgriezta veikalā. Atgrieztais daudzums: 30

Jūsu izvelētas preces:

Āboli - cena: 1.00€ , daudzums: 20

Pipari - cena: 1.00€ , daudzums: 66

Kļūda! Nevar atgriezt negatīvu daudzumu!

Jūsu izvelētas preces:

Āboli - cena: 1.00€ , daudzums: 20

Pipari - cena: 1.00€ , daudzums: 66

Kļūda! Nevar atgriezt neko!

Jūsu izvelētas preces:

Āboli - cena: 1.00€ , daudzums: 20

Pipari - cena: 1.00€ , daudzums: 66

---------- ČEKS ----------

SIA "MAKSIMA" Latvija

Veikals "Maksima"

Dārza iela 21, Bauska, t.22813371

Juridiskā adrese: "Abras", Krustkalni,

Ķekavas pagasts, Ķekavas novads, LV-2222

Čeks 442/706 #80908615

==========

Āboli - cena: 1.00€, daudzums: 20, 20.00€

Pipari - cena: 1.00€, daudzums: 66, 66.00€

==========

Kopā apmaksai: 86.00€

Nopelnīta "MAKSIMA" nauda: 0.86€

Pieejamas preces veikalā:

Āboli - cena: 1.00€ , daudzums: 30

Olas - cena: 1.50€ , daudzums: 30

Pipari - cena: 1.00€ , daudzums: 34

2)

```
Pieejamas preces veikalā:
Āboli - cena: 1.00€ , daudzums: 50
Piens - cena: 1.40€ , daudzums: 20
Olas - cena: 1.50€ , daudzums: 30
Pipari - cena: 1.00€ , daudzums: 100

Prece "Piens" tika izmesta Getliņu izgāztuvē.

Pieejamas preces veikalā:
Āboli - cena: 1.00€ , daudzums: 50
Olas - cena: 1.50€ , daudzums: 30
Pipari - cena: 1.00€ , daudzums: 100

Kļūda! Veikalā nav tik daudz preču! Precei "Olas" pieejamais daudzums: 30
Izvēlētas preces: Olas, cena: 1.50€ , daudzums: 30
Kļūda! Prece "Vārdnīca" nav pieejama veikalā!
Izvēlētas preces: Āboli, cena: 1.00€ , daudzums: 50
Izvēlētas preces: Pipari, cena: 1.00€ , daudzums: 66

Jūsu izvelētas preces:
Olas - cena: 1.50€ , daudzums: 30
Āboli - cena: 1.00€ , daudzums: 50
Pipari - cena: 1.00€ , daudzums: 66

Izvēlētā prece "Āboli" tika atgriezta veikalā. Atgrieztais daudzums: 30
Izvēlētā prece "Olas" tika atgriezta veikalā. Atgrieztais daudzums: 30

Jūsu izvelētas preces:
Āboli - cena: 1.00€ , daudzums: 20
Pipari - cena: 1.00€ , daudzums: 66

Kļūda! Nevar atgriezt negatīvu daudzumu!
```

3)

```
Jūsu izvelētas preces:
Āboli - cena: 1.00€ , daudzums: 20
Pipari - cena: 1.00€ , daudzums: 66

Kļūda! Nevar atgriezt neko!
Jūsu izvelētas preces:
Āboli - cena: 1.00€ , daudzums: 20
Pipari - cena: 1.00€ , daudzums: 66


--------------- ČEKS ---------------
SIA "MAKSIMA" Latvija
Veikals "Maksima"
Dārza iela 21, Bauska, t.22813371
Juridiskā adrese: "Abras", Krustkalni,
Ķekavas pagasts, Ķekavas novads, LV-2222
Čeks 324/956                    #49612886
==========================================
Āboli - cena: 1.00€, daudzums: 20, 20.00€
Pipari - cena: 1.00€, daudzums: 66, 66.00€
==========================================
Kopā apmaksai: 86.00€
Nopelnīta "MAKSIMA" nauda: 0.86€
------------------------------------------


Pieejamas preces veikalā:
Āboli - cena: 1.00€ , daudzums: 30
Olas - cena: 1.50€ , daudzums: 30
Pipari - cena: 1.00€ , daudzums: 34
```