14. НАСЛІДУВАННЯ ТА АГРЕГУВАННЯ

Т14.1 Клас Point2 описано наступним чином:

```
class Point2:
    """Клас реалізує точку площини"""
    def __init__(self, x, y):
        self._x = x
        self._y = y

    def get_x(self):
        """Повернути координату x"""
        return self._x

    def get_y(self):
        """Повернути координату y"""
        return self._y

    def __str__(self):
        """Повернути рядок представлення точки"""
        return "({}, {})".format(self._x, self._y)
```

Описати клас Rectangle – прямокутник, заданий 4 вершинами. Реалізувати методи повернення вершин, обчислення периметру та площі, __str__.
Використати цей клас для розв'язання задачі. Задано декілька прямокутників, сторони яких паралельні осям координат.
Знайти площу
а) перетину цих прямокутників;
б) об'єднання цих прямокутників.

Т14.2 Клас Point2 описано у задачі Т14.1. Описати клас RegularPolygone - правильний многокутник. Правильний многокутник задається списком вершин (точок). Реалізувати методи повернення вершин, обчислення периметру та площі, __str__.
Використати цей клас для розв'язання задачі. Задано декілька правильних многокутників. Перевірити, чи є серед них такий, який охоплює всі інші многокутники (тобто всі вершини всіх інших многокутників лежать всередині цього многокутника).

Т14.3 Використати стандартний контейнер Counter для розв'язання задачі. Дана непорожня послідовність символів (рядок) *S*. Описати функцію, яка визначає загальну кількість цифр та знаків операцій +, -, *. /, що входять до рядка *S*.

Т14.4 Використати стандартний контейнер Counter для розв'язання задач.

а) знайти символ, який входить у рядок *S* максимальну кількість разів;
б) перевірити, чи складаються рядки *S1*, *S2* з одних і тих же символів, які входять у ці рядки однакову кількість разів;
в) перевірити, чи вірно, що всі символи рядка *S1*, входять також у рядок *S2*, причому не меншу кількість разів, ніж у *S1*;
г) перевірити, чи вірно, що жоден символ рядка *S1* не входить у рядок *S2*, та жоден символ *S2* не входить до *S1*.

Т14.5 Ненульові елементи розрідженої матриці зберігаються у словнику (defaultdict). Ключами словника є кортежі, що складаються з індексів рядка та стовпчика, а значеннями словника, - значення елементів матриці. Обчислити:
а) суму двох розріджених матриць однакового розміру

б) добуток двох розріджених матриць

в) значення мінімального елемента розрідженої матриці

г) значення максимального елемента розрідженої матриці

д) добуток розрідженої матриці на розріджений вектор (зберігається також у словнику)

е) чи є розріджена матриця нижньою трикутною

є) чи є розріджена матриця діагональною

Т14.6 За допомогою стандартного контейнера deque розв'язати задачу:
По колу розташовано n гравців з номерами від 1 до n. У лічилці m слів. Починають лічити з першого гравця. m-й за ліком вибуває. Потім знову лічать з наступного гравця за вибулим. Знову m-й вибуває. Так продовжують, поки не залишиться жодного гравця. Треба показати послідовність номерів, що вибувають, при заданих n та m.

Т14.7 За допомогою стандартного контейнера deque розв'язати задачу:
У магазині стоїть черга з *m* покупців. Час обслуговування покупця з черги – це випадкове ціле число в діапазоні від 1 до t_1. Час додавання нового покупця до черги - це випадкове ціле число в діапазоні від 1 до t_2. Промоделювати стан черги (тобто показати час виникнення подій – обслуговування та додавання покупця) за період часу T ($T >> t_1$, $T >> t_2$). Показати залишок черги.

Т14.8 Розв'язати задачу Т14.7, передбачивши що через випадковий час від 1 до t_3 до початку черги додається „пільговий" покупець, який обслуговується першим, а через випадковий час від 1 до t_4 не витримує та йде з черги останній покупець.

T14.9 За допомогою стандартного контейнера deque розв'язати задачу:
Є невпорядкована послідовність дійсних чисел. Ці числа треба розмістити у контейнері deque та, щоб числа буди впорядковані за неспаданням, починаючи від початку deque.
sort або інші способи сортування списків не використовувати.

T14.9 За допомогою стандартного контейнера deque розв'язати задачу:
Є невпорядкована послідовність дійсних чисел. Ці числа треба розмістити у контейнері deque так, щоб числа буди впорядковані за неспаданням, починаючи від початку deque.
sort або інші способи сортування списків не використовувати.

T14.10 Клас Point2 описано у задачі T14.1
Описати клас Point2Ex як нащадок Point2. У цьому класі реалізувати особливі методи для відношень <, == тощо. Вважати що точки впорядковані за відстанню від початку координат.
Використати клас Point2Ex для розв'язання задачі: знайти довжину ламаної лінії, яка проведена між послідовністю точок, які впорядковано за вдіношенням "<".

T14.11 Клас Segment, який використовує клас Point2, описано наступним чином

```
class Segment:
    """Клас реалізує відрізок на площині"""

    def __init__(self, a, b):
        self._a = a      # точка
        self._b = b      # точка

    def get_a(self):
        """Повернути точку a"""
        return self._a

    def get_b(self):
        """Повернути точку b"""
        return self._b

    def __str__(self):
        """Повернути рядок представлення відрізку"""
        return "[{}, {}]".format(self._a, self._b)

    def len(self):
        return sqrt((self._a.get_x() - self._b.get_x()) ** 2 +
                    (self._a.get_y() - self._b.get_y()) ** 2)
```

Описати клас SegmentEx як нащадок Segment. У цьому класі реалізувати особливі методи для відношень <, == тощо. Вважати що відрізки впорядковані за їх довжиною.
Ввести декілька відрізків та показати їх у порядку впорядкування.

T14.12 Клас Triangle, який використовує клас Point2, описано наступним чином

```
class Triangle:
    """Клас реалізує трикутник"""

    def __init__(self, a, b, c):
        self._a = a      # точка
        self._b = b      # точка
        self._c = c      # точка

    def get_a(self):
        """Повернути точку a"""
        return self._a

    def get_b(self):
        """Повернути точку b"""
        return self._b

    def get_c(self):
        """Повернути точку c"""
        return self._c

    def __str__(self):
        """Повернути рядок представлення трикутника"""
        return "Трикутник ({}, {}, {})".format(self._a, self._b,
self._c)

    def _get_sides(self):
        s1 = Segment(self._a, self._b).len()
        s2 = Segment(self._b, self._c).len()
        s3 = Segment(self._a, self._c).len()
        return s1, s2, s3

    def perimeter(self):
        """Повернути периметр трикутника"""
        s1, s2, s3 = self._get_sides()
        return s1 + s2 + s3

    def square(self):
        p = self.perimeter() / 2
        s1, s2, s3 = self._get_sides()
        return sqrt(p * (p - s1) * (p - s2) * (p - s3))
```

Описати клас TriangleEx як нащадок Triangle. У цьому класі реалізувати особливі методи для відношень <, == тощо. Вважати що трикутники впорядковані за їх площею.
Ввести декілька трикутників та показати їх у порядку впорядкування.

Т14.13 Описати клас для роботи з відрізками на числовій осі. Для відрізку передбачити поля:

(*a, b, empty)*

де *a, b* - границі відрізку, *empty* - ознака того, що відрізок порожній.
Реалізувати методи:

1) зробити відрізок *t* порожнім;
2) чи порожній відрізок *t*;
3) покласти відрізок *t* рівним *a, b*;
4) покласти відрізок *t* рівним перетину відрізків *t1, t2*;
5) описати особливий метод для відношення in (__contains__(self, other)), який перевіряє, чи містить один відрізок інший.
6) реалізувати особливі методи для відношень <, == тощо. Вважати що відрізки впорядковані за їх довжиною.

З використанням класу скласти програму, яка вводить декілька відрізків та перевіряє, чи існує відрізок, який містить усі інші.

Т14.14 Описати клас для реалізації мультимножини цілих чисел на базі словника. Мультимножина - це множина в якій для кожного елемента запам'ятовується не лише його входження, але й кількість входжень.

Кількість входжень елемента *k* (1<=*k*<=*n*) у мультимножину - це значення елемента словника з ключем *k*.
Реалізувати методи:

1) зробити мультимножину порожньою;
2) чи є мультимножина порожньою;
3) додати елемент до мультимножини;
4) забрати елемент з мультимножини (кількість входжень елемента зменшується на 1, якщо елемент не входить - відмова);
5) кількість входжень елемента у мультимножину - реалізувати особливий метод __getitem__;
6) об'єднання двох мультимножин (в результаті об'єднання кідькість входжень елемента визначається як максимальна з двох мультимножин) – реалізувати особливий метод для операції | (вертикальна риска) __or__(self, other);
7) перетин двох мультимножин (в результаті кількість входжень елемента визначається як мінімальна з двох мультимножин) – реалізувати особливий метод для операції & __and__(self, other);

З використанням класу розв'язати задачі:
а) знайти символ, який входить у рядок *S* максимальну кількість разів;

б) перевірити, чи складаються рядки *S1*, *S2* з одних і тих же символів, які входять у ці рядки однакову кількість разів;
в) перевірити, чи вірно, що всі символи рядка *S1*, входять також у рядок *S2*, причому не меншу кількість разів, ніж у *S1*.

T14.15 В умовах задачі T14.14 реалізувати у класі для мультимножин особливі методи для відношень <, == тощо. Вважати що відношеня < позначає включення однієї мультимножини у іншу.
З використанням класу розв'язати задачу. Ввести послідовність мультимножин та перевірити, чи є вона лінійно впорядкованою за вдношенням "<"

T14.16 Клас Point2 описано у задачі T14.1
Описати клас Point2Ex як нащадок Point2. У цьому класі реалізувати особливі методи для відношень ==, !=. Вважати що точки рівні, якщо рівні їх координати.
Клас Segment, який використовує клас Point2, описано у задачі 14.11
Описати клас SegmentEx, як нащадок Segment. У цьому класі реалізувати особливі методи для відношень <, == тощо. Вважати що відрізки впорядковані за довжиною. Також реалізувати метод, що перевіряє, чи лежить відрізок на одній прямій з іншим відрізком.

Використати класи Point2Ex, SegmentEx для розв'язання задачі: задано декілька відрізків. Перевірити, чи складають вони многокутник та чи є він правильним.