ПРЕЗЕНТАЦИЮ ПОДГОТОВИЛИ:

СТУДЕНТЫ ГР. 5130901/20201
К.М. Зезина
Н.А. Дюков
СТУДЕНТЫ ГР. 5130901/20101
А. Теллили
А.Д. Усачев
СТУДЕНТЫ ГР. 5130901/20102
О.С. Соловьев
А.А. Вагнер
СТУДЕНТЫ ГР. 5130901/20103
С.А. Мукий
Ф.Г. Кудрин

РАЗРАБОТКА БИБЛИОТЕКИ ФУНКЦИЙ ДЛЯ ОБРАБОТКИ СИМВОЛЬНЫХ ПРЕДСТАВЛЕНИЙ МАТЕМАТИЧЕСКИХ ОБЪЕКТОВ, ИМЕЮЩИХ АЛГЕБРАИЧЕСКУЮ СТРУКТУРУ - КОЛЬЦО

СТРУКТУРА АБЕЛЕВОЙ ГРУППЫ

Свойства сложения:

- $a + b = b + a$ (коммутативность)
- $(a + b) + c = a + (b + c)$ (ассоциативность)
- $a + 0 = a$
- $a + (-a) = 0$

Свойства умножения:

- $ab = ba$ (коммутативность)
- $(ab)c = a(bc)$ (ассоциативность)
- $a1 = a$
- $aa^{-1} = 1$ при $a \neq 0$

ЧТО ТАКОЕ КОЛЬЦО?

Кольцо — множество **K** с операциями сложения и умножения, обладающее следующими свойствами:

- K ЯВЛЯЕТСЯ АБЕЛЕВОЙ ГРУППОЙ ПО СЛОЖЕНИЮ (АДДИТИВНАЯ ГРУППА).
- УМНОЖЕНИЕ ДИСТРИБУТИВНО ОТНОСИТЕЛЬНО СЛОЖЕНИЯ: $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$

СЛЕДСТВИЯ АКСИОМ КОЛЬЦА

- **Для любого a:** $a \cdot 0 = 0$
- **Для любых A И B:** $A(-B) = (-A)B$
- **Умножение дистрибутивно в обе стороны:**
  $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$ И АНАЛОГИЧНО С ДРУГОЙ СТОРОНЫ.
- **Умножение в кольце K:**
  - *КОММУТАТИВНО*, ЕСЛИ $A \cdot B = B \cdot A$
  - *АССОЦИАТИВНО*, ЕСЛИ $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

ПРИМЕРЫ АЛГЕБРАИЧЕСКИХ КОЛЕЦ

- ЧИСЛОВЫЕ МНОЖЕСТВА ($\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$) — КОММУТАТИВНЫЕ, АССОЦИАТИВНЫЕ КОЛЬЦА С ЕДИНИЦЕЙ.
- ЧЕТНЫЕ ЧИСЛА ($2\mathbb{Z}$)— КОММУТАТИВНОЕ, АССОЦИАТИВНОЕ КОЛЬЦО БЕЗ ЕДИНИЦЫ.
- МНОЖЕСТВО ФУНКЦИЙ НА ЧИСЛОВОЙ ПРЯМОЙ — КОММУТАТИВНОЕ КОЛЬЦО С ЕДИНИЦЕЙ.
- МНОЖЕСТВО ВЕКТОРОВ С ОПЕРАЦИЯМИ СЛОЖЕНИЯ И ВЕКТОРНОГО УМНОЖЕНИЯ — НЕКОММУТАТИВНОЕ И НЕАССОЦИАТИВНОЕ КОЛЬЦО.

СТРУКТУРА И РЕАЛИЗАЦИЯ БИБЛИОТЕКИ

ЧАСТЬ 1. СОЗДАНИЕ И УДАЛЕНИЕ ОБЪЕКТА

```
# Создание
def __init__(self, coefficients):
    self.coefficients = coefficients
    self.reduce()

# Удаление
def delete(self):
    self.coefficients = None
```

В ФУНКЦИЮ ПОДАЮТСЯ ЗНАЧЕНИЯ ОДНОГО ПОЛИНОМА: В ПОРЯДКЕ ИНДЕКСА – СТЕПЕНЬ ПРИ НИХ

```
p1 = Ring([1, 2])  # 2x + 1
p2 = Ring([3, 1])  # x + 3
p3 = Ring([0, 1])  # x
```

ЧАСТЬ 2. РЕДУКЦИЯ (УПРОЩЕНИЕ) ВЫРАЖЕНИЙ, СОДЕРЖАЩИХ ОБЪЕКТЫ И ИХ СВОЙСТВА

```
# Убирает лишние нули если есть
def reduce(self):
      while self.coefficients and self.coefficients[-1] == 0
          self.coefficients.pop()
return self
```

Пример:

```
p4 = Ring([0, 0, 0, 3, 0, 0]) # 3x^3 + 0x^2 + 0x + 0
```

Результат:

```
3x^3
```

ЧАСТЬ 3. КОПИРОВАНИЕ ОБЪЕКТА

```
from copy import deepcopy

#Копирование текущего объекта кольца.
# return: Новый объект кольца, идентичный текущему.

def copy(self):
      return deepcopy(self)
```

Пример:

```
r = [1, 2, 3]
r.append(r)
print(r) # [1, 2, 3, [1, 2, 3]]
p = copy.deepcopy(r)
print(p) # [1, 2, 3, [1, 2, 3]]
```

ЧАСТЬ 4. ПОСТРОЕНИЕ ВЫРАЖЕНИЙ, СОДЕРЖАЩИХ ОБЪЕКТОВ И ИХ СВОЙСТВА

```
# Сложение
def __add__(self, other):
    max_len = max(len(self.coefficients),
len(other.coefficients))
    padded_self = self.coefficients + [0] * (max_len -
len(self.coefficients))
    padded_other = other.coefficients + [0] * (max_len -
len(other.coefficients))
    result_coeffs = [a + b for a, b in zip(padded_self,
padded_other)]
    return Ring(result_coeffs)

# Умножение
def __mul__(self, other):
    result_coeffs = [0] * (len(self.coefficients) +
len(other.coefficients) - 1)
    for i, a in enumerate(self.coefficients):
        for j, b in enumerate(other.coefficients):
            result_coeffs[i + j] += a * b
    return Ring(result_coeffs)

# Сравнение
def __eq__(self, other):
    # Приводим коэффициенты к одинаковой длине
    max_len = max(len(self.coefficients),
len(other.coefficients))
    padded_self = self.coefficients + [0] * (max_len -
len(self.coefficients))
    padded_other = other.coefficients + [0] * (max_len -
len(other.coefficients))
    return padded_self == padded_other
```

ПРОВЕРКА СВОЙСТВ

```
1. Проверка коммутативности сложения
p1 + p2 = p2 + p1  => 4 + 3x = 4 + 3x
Свойство коммутативности сложения не нарушено

2. Проверка ассоциативности сложения
(p1 + p2) + p3 = p1 + (p2 + p3)  => 4 + 4x = 4 + 4x
Свойство ассоциативности сложения не нарушено

3. Проверка существования нуля
p1 * p0 = p0  => 0 = 0
Свойство существования нуля не нарушено

4. Проверка коммутативности умножения
p1 * p2 = p2 * p1  => 3 + 7x + 2x^2 = 3 + 7x + 2x^2
Свойство коммутативности умножения не нарушено
```

```
5. Проверка ассоциативности умножения
(p1 * p2) * p3 = p1 * (p2 * p3)  => 3x + 7x^2 + 2x^3 = 3x + 7x^2 + 2x^3
Свойство ассоциативности умножения не нарушено

6. Проверка дистрибутивности
p1 * (p2 + p3) = (p1 * p2) + (p1 * p3)  => 3 + 8x + 4x^2 = 3 + 8x + 4x^2
Свойство дистрибутивности не нарушено

7.  Проверка существования единицы
 p1 * one = p1  => 1 + 2x = 1 + 2x
Свойство существования единицы не нарушено
```

ЧАСТЬ 5. ОПРЕДЕЛЕНИЕ И ПЕРЕОПРЕДЕЛЕНИЕ СВОЙСТВ ОБЪЕКТА

```
# Преобразование в строку
def __str__(self):
    terms = []
    for i, coeff in enumerate(self.coefficients):
        if coeff != 0:
            if coeff == 1 and i > 0:
                coeff_str = ""
            elif coeff == -1 and i > 0:
                coeff_str = "-"
            else:
                coeff_str = str(coeff)
            if i == 0:
                terms.append(coeff_str)
            elif i == 1:
                terms.append(f"{coeff_str}x")
            else:
                terms.append(f"{coeff_str}x^{i}")
    if not terms:
        return "0"
    return " + ".join(terms)
```

СПАСИБО ЗА ВНИМАНИЕ!

Мы готовы ответить
на ваши вопросы ☺