Ответ на задание 1 вариант 1

```
class Food:
    def __init__(self, proteins: int, fats: int, carbohydrates: int):
        """Инициализация объекта Food с указанием белков, жиров и углеводов (в граммах)."""
        self.proteins = proteins  # Количество белков (г)
        self.fats = fats          # Количество жиров (г)
        self.carbohydrates = carbohydrates  # Количество углеводов (г)

    def get_proteins(self) -> int:
        """Возвращает количество белков в граммах."""
        return self.proteins

    def get_fats(self) -> int:
        """Возвращает количество жиров в граммах."""
        return self.fats

    def get_carbohydrates(self) -> int:
        """Возвращает количество углеводов в граммах."""
        return self.carbohydrates

    def get_calories(self) -> int:
        """Вычисляет калорийность по формуле: 4*белки + 9*жиры + 4*углеводы."""
        return 4 * self.proteins + 9 * self.fats + 4 * self.carbohydrates

    def __add__(self, other):
        """Перегрузка оператора + для сложения двух объектов Food.
        Возвращает новый объект Food с суммарными БЖУ."""
        if not isinstance(other, Food):
            raise TypeError("Можно складывать только объекты Food!")

        # Суммируем компоненты
        new_proteins = self.proteins + other.proteins
        new_fats = self.fats + other.fats
```

```
        new_carbohydrates = self.carbohydrates + other.carbohydrates

        # Создаем новый объект с суммарными значениями
        return Food(new_proteins, new_fats, new_carbohydrates)

    def __repr__(self):
        """Строковое представление объекта (для print и отладки)."""
        return f"Food(proteins={self.proteins}, fats={self.fats}, carbohydrates={self.carbohydrates})"
```

Пример применение

```
apple = Food(0, 0, 25)  # Яблоко: 0 белков, 0 жиров, 25 углеводов
chicken = Food(26, 3, 0)  # Курица: 26 белков, 3 жира, 0 углеводов

print(apple.get_calories())  # 100 ккал (0*4 + 0*9 + 25*4)
print(chicken.get_calories())  # 131 ккал (26*4 + 3*9 + 0*4)

meal = apple + chicken  # Складываем два продукта
print(meal)  # Food(proteins=26, fats=3, carbohydrates=25)
print(meal.get_calories())  # 221 ккал (26*4 + 3*9 + 25*4)
```

Задание 2 вариант 1

```
class IntVector:
    def __init__(self, data):
        """Инициализация вектора с заданными данными"""
        self.data = list(data)  # Храним данные как список

    def __len__(self):
        """Возвращает длину вектора"""
        return len(self.data)

    def __getitem__(self, index):
        """Обращение к элементу по индексу с проверкой границ"""
        if index < 0 or index >= len(self.data):
            raise IndexError("Индекс выходит за границы массива")
        return self.data[index]

    def __setitem__(self, index, value):
        """Установка значения элемента по индексу с проверкой границ"""
        if index < 0 or index >= len(self.data):
            raise IndexError("Индекс выходит за границы массива")
        self.data[index] = value

    def __add__(self, other):
        """Поэлементное сложение векторов одинаковой длины"""
        if len(self) != len(other):
            raise ValueError("Векторы должны быть одинаковой длины")
        return IntVector([a + b for a, b in zip(self.data, other.data)])

    def __sub__(self, other):
        """Поэлементное вычитание векторов одинаковой длины"""
        if len(self) != len(other):
            raise ValueError("Векторы должны быть одинаковой длины")
        return IntVector([a - b for a, b in zip(self.data, other.data)])
```

```
    def __mul__(self, scalar):
        """Умножение всех элементов вектора на скаляр"""
        return IntVector([x * scalar for x in self.data])

    def __truediv__(self, scalar):
        """Деление всех элементов вектора на скаляр (целочисленное)"""
        if scalar == 0:
            raise ZeroDivisionError("Деление на ноль")
        return IntVector([x // scalar for x in self.data])

    def print_element(self, index):
        """Вывод элемента по индексу"""
        print(f"Элемент с индексом {index}: {self[index]}")

    def print_all(self):
        """Вывод всего вектора"""
        print("Вектор:", self.data)

    def __repr__(self):
        """Строковое представление вектора"""
        return f"IntVector({self.data})"


# Демонстрация работы класса
if __name__ == "__main__":
    # Создаем векторы
    v1 = IntVector([1, 2, 3, 4, 5])
    v2 = IntVector([5, 4, 3, 2, 1])

    print("Вектор 1:", v1)
    print("Вектор 2:", v2)
```

```
print()

# Тестируем доступ по индексу
try:
    print("Элемент v1[2]:", v1[2])
    print("Элемент v1[10]:", v1[10])  # Должно вызвать исключение
except IndexError as e:
    print("Ошибка:", e)
print()

# Тестируем операции
print("Сумма векторов:", v1 + v2)
print("Разность векторов:", v1 - v2)
print("Умножение v1 на 2:", v1 * 2)
print("Деление v2 на 2:", v2 / 2)
print()

# Тестируем вывод
v1.print_element(3)
v2.print_element(0)
v1.print_all()
v2.print_all()
```