```
from operator import itemgetter
import unittest

class Emp:
    """Сотрудник"""
    def __init__(self, id, fio, sal, dep_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.dep_id = dep_id

class Dep:
    """Отдел"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class EmpDep:
    """
    'Сотрудники отдела' для реализации
    связи многие-ко-многим
    """
    def __init__(self, dep_id, emp_id):
        self.dep_id = dep_id
        self.emp_id = emp_id

# Отделы
deps = [
    Dep(1, 'отдел кадров'),
    Dep(2, 'архивный отдел ресурсов'),
    Dep(3, 'бухгалтерия'),
    Dep(11, 'отдел (другой) кадров'),
```

```
    Dep(22, 'архивный (другой) отдел ресурсов'),
    Dep(33, '(другая) бухгалтерия'),
]

# Сотрудники
emps = [
    Emp(1, 'Артамонов', 25000, 1),
    Emp(2, 'Пронин', 35000, 2),
    Emp(3, 'Осада', 45000, 3),
    Emp(4, 'Папин', 35000, 3),
    Emp(5, 'Заира', 25000, 3),
]

emps_deps = [
    EmpDep(1,1),
    EmpDep(2,2),
    EmpDep(3,3),
    EmpDep(3,4),
    EmpDep(3,5),
    EmpDep(11,1),
    EmpDep(22,2),
    EmpDep(33,3),
    EmpDep(33,4),
    EmpDep(33,5),
]


def task_1_one_to_many(deps, emps):
    """Выполняет Задание 1: связывает сотрудников и отделы один-ко-многим."""
    one_to_many = [(e.fio, e.sal, d.name) for d in deps for e in emps if e.dep_id == d.id]
    return sorted(one_to_many, key=itemgetter(2))
```

```
def task_2_department_salaries(deps, emps):
    """Выполняет Задание 2: Суммирует зарплаты по отделам."""
    dep_salaries = {}
    for d in deps:
        total_salary = sum(e.sal for e in emps if e.dep_id == d.id)
        dep_salaries[d.name] = total_salary
    return sorted(dep_salaries.items(), key=itemgetter(1), reverse=True)


def task_3_many_to_many(deps, emps, emps_deps):
    """Выполняет Задание 3: Связывает сотрудников и отделы многие-ко-многим."""
    dep_emp_map = {}
    for ed in emps_deps:
        dep = next((d for d in deps if d.id == ed.dep_id), None)
        emp = next((e for e in emps if e.id == ed.emp_id), None)
        if dep and emp:
            dep_name = dep.name
            emp_name = emp.fio
            if dep_name not in dep_emp_map:
                dep_emp_map[dep_name] = set()
            dep_emp_map[dep_name].add(emp_name)

    return {dep: sorted(list(emps)) for dep, emps in dep_emp_map.items()}


def main():
    """Основная функция, которая вызывает все задания и выводит результаты."""

    # Задание 1:
    print('Задание А1')
    res_11 = task_1_one_to_many(deps, emps)
```

```
    print(res_11)

    # Задание 2:
    print('\nЗадание A2')
    res_12 = task_2_department_salaries(deps, emps)
    print(res_12)

    # Задание 3:
    print('\nЗадание A3')
    res_13 = task_3_many_to_many(deps, emps, emps_deps)
    print(res_13)


class TestTasks(unittest.TestCase):
    """Набор тестов для проверки функций заданий."""

    def test_task_1_one_to_many(self):
        """Тест для Задания 1."""
        expected_result = [
            ('Артамонов', 25000, 'архивный отдел ресурсов'),
            ('Пронин', 35000, 'бухгалтерия'),
            ('Осада', 45000, 'отдел кадров'),
            ('Папин', 35000, 'бухгалтерия'),
            ('Заира', 25000, 'бухгалтерия'),
        ]
        actual_result = task_1_one_to_many(deps, emps)
        self.assertEqual(actual_result, expected_result)


    def test_task_2_department_salaries(self):
        """Тест для Задания 2."""
        expected_result = [
```

```
            ('бухгалтерия', 105000),
            ('архивный отдел ресурсов', 35000),
            ('отдел кадров', 25000),
            ('архивный (другой) отдел ресурсов', 0),
            ('(другая) бухгалтерия', 0),
            ('отдел (другой) кадров', 0)

        ]
        actual_result = task_2_department_salaries(deps, emps)
        self.assertEqual(actual_result, expected_result)

    def test_task_3_many_to_many(self):
        """Тест для Задания 3."""
        expected_result = {
            'отдел кадров': ['Артамонов'],
            'архивный отдел ресурсов': ['Пронин'],
            'бухгалтерия': ['Заира', 'Осада', 'Папин'],
            'отдел (другой) кадров': ['Артамонов'],
            'архивный (другой) отдел ресурсов': ['Пронин'],
            '(другая) бухгалтерия': ['Заира', 'Осада', 'Папин']
        }
        actual_result = task_3_many_to_many(deps, emps, emps_deps)
        self.assertEqual(actual_result, expected_result)

if __name__ == '__main__':
    main()
    unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

Результат :

Задание А1

[('Артамонов', 25000, 'архивный отдел ресурсов'), ('Пронин', 35000, 'бухгалтерия'), ('Осада', 45000, 'отдел кадров'), ('Папин', 35000, 'бухгалтерия'), ('Заира', 25000, 'бухгалтерия')]

Задание A2

[('бухгалтерия', 105000), ('архивный отдел ресурсов', 35000), ('отдел кадров', 25000), ('архивный (другой) отдел ресурсов', 0), ('(другая) бухгалтерия', 0), ('отдел (другой) кадров', 0)]

Задание A3

{'отдел кадров': ['Артамонов'], 'архивный отдел ресурсов': ['Пронин'], 'бухгалтерия': ['Заира', 'Осада', 'Папин'], 'отдел (другой) кадров': ['Артамонов'], 'архивный (другой) отдел ресурсов': ['Пронин'], '(другая) бухгалтерия': ['Заира', 'Осада', 'Папин']}

.

Ran 3 tests in 0.000s