Код программы:

```
from operator import itemgetter

class Disk:
    # CD-диск
    def __init__(self, id, name, memory, lib_id):
        self.id = id
        self.name = name
        self.memory = memory
        self.lib_id = lib_id

class Lib:
    # библиотека с CD-дисками
    def __init__(self, id, name):
        self.id = id
        self.name = name

class DiskLib:
    # CD-диски библиотеки
    # для реализации связи многие-ко-многим
    def __init__(self, lib_id, disk_id):
        self.lib_id = lib_id
        self.disk_id = disk_id

# библиотеки с CD-дисками
libs = [
Lib(1, "Музыкальная библиотека"),
Lib(2, "Архив поэзии"),
Lib(3, "Архив фото"),
Lib(11, "Музыкальная библиотека (другая)"),
Lib(22, "Архив поэзии (другой)"),
Lib(33, "Архив фото (другой)"),
]

# CD-диски
disks = [
Disk(1, "Группа Кино", 650, 1),
Disk(2, "Сборник Маяковского", 650, 2),
Disk(3, "Фото с отдыха", 700, 3),
Disk(4, "Классическая музыка", 700, 1),
Disk(5, "Рок", 600, 1),
Disk(6, "Фото с праздника", 750, 3),
]

disks_libs = [
DiskLib(1, 1),
DiskLib(2, 2),
DiskLib(3, 3),
DiskLib(1, 4),
DiskLib(1, 5),
```

```
DiskLib(3, 6),
DiskLib(11, 1),
DiskLib(22, 2),
DiskLib(33, 3),
DiskLib(11, 4),
DiskLib(11, 5),
DiskLib(33, 6),
]

def main():
    # соединение данных один-ко-многим
    one_to_many = [(d.name, d.memory, l.name)
        for l in libs
        for d in disks
        if d.lib_id == l.id]
    # соединение данных многие-ко-многим
    many_to_many_temp = [(l.name, dl.lib_id, dl.disk_id)
        for l in libs
        for dl in disks_libs
        if l.id == dl.lib_id]

    many_to_many = [(d.name, d.memory, lib_name)
        for lib_name, lib_id, disk_id in many_to_many_temp
        for d in disks if d.id == disk_id]

    print("Задание Г1")
    res_1 = {}
    # перебираем библиотеки
    for l in libs:
        if 'A' == l.name[0]:
            # формируем список CD-дисков в библиотеке
            l_disks = list(filter(lambda i : i[2] == l.name, one_to_many))
            # если библиотека не пуста
            if len(l_disks) > 0:
                # оставляем только названия дисков
                l_disks_names = [n for n,_,_ in l_disks]
                res_1[l.name] = l_disks_names
    print(res_1)

    print("\nЗадание Г2")
    res_2_unsorted = []
    # перебираем все библиотеки
    for l in libs:
        # формируем список CD-дисков в библиотеке
        l_disks = list(filter(lambda i : i[2] == l.name, one_to_many))
        # если библиотека не пуста
        if len(l_disks) > 0:
            # формируем список значений памяти CD-дисков в библиотеке
            l_memory = [m for _,m,_ in l_disks]
            # берем максимальную память CD-диска в библиотеке
            l_max_memory = max(l_memory)
```

```
            res_2_unsorted.append((l.name, l_max_memory))
    # сортировка по максимальной памяти CD-диска
    res_2 = sorted(res_2_unsorted, key = itemgetter(1), reverse = True)
    print(*res_2, sep='\n')


    print("\nЗадание Г3")
    res_3 = sorted(many_to_many, key = itemgetter(2))
    print(*res_3, sep='\n')


if __name__ == "__main__":
    main()
```

Результаты выполнения программы:

Задание Г1

{'Архив поэзии': ['Сборник Маяковского'], 'Архив фото': ['Фото с отдыха', 'Фото с праздника']}

Задание Г2

('Архив фото', 750)

('Музыкальная библиотека', 700)

('Архив поэзии', 650)

Задание Г3

('Сборник Маяковского', 650, 'Архив поэзии')

('Сборник Маяковского', 650, 'Архив поэзии (другой)')

('Фото с отдыха', 700, 'Архив фото')

('Фото с праздника', 750, 'Архив фото')

('Фото с отдыха', 700, 'Архив фото (другой)')

('Фото с праздника', 750, 'Архив фото (другой)')

('Группа Кино', 650, 'Музыкальная библиотека')

('Классическая музыка', 700, 'Музыкальная библиотека')

('Рок', 600, 'Музыкальная библиотека')

('Группа Кино', 650, 'Музыкальная библиотека (другая)')

('Классическая музыка', 700, 'Музыкальная библиотека (другая)')

('Рок', 600, 'Музыкальная библиотека (другая)')