**from** random **import** randint  
  
UNIC\_NAMES\_LIST = [  
 **"Комова Елизавета Олеговна"**,  
 **"Репенков Сергей Алексеевич"**,  
 **"Анисимов Ефим Сергеевич"**,  
 **"Борисов Никита Алексеевич"**,  
 **"Исмоилова Милена Витальевна"**,  
 **"Рогов Владимир Алексеевич"**,  
 **"Савочкин Артём Дмитриевич"**,  
 **"Кривошеина Елена Олеговна"**,  
 **"Пономарев Михаил Александрович"**,  
 **"Абдулмеджидов Мирза Мурадович"**,  
 **"Егшатян Артем Кирович"**,  
 **"Гарайшин Тамерлан Тагирович"**,  
 **"Брусова Полина Игоревна"**,  
 **"Крылова Елизавета Алексеевна"**,  
 **"Гераськина Надежда Станиславовна"**,  
 **"Гиниятуллина Эвита Маратовна"**,  
 **"Малахов Иван Петрович"**,  
 **"Жилина Алена Алексеевна"**,  
 **"Королев Илья Алексеевич"**,  
 **"Пойкалайнен Александра Максимовна"**,  
 **"Щербак Станислав Валентинович"**,  
 **"Буркина Елизавета Сергеевна"**,  
 **"Мосолова Ксения Дмитриевна"**,  
 **"Кротов Олег Валерьевич"**,  
 **"Шаповалов Сергей Александрович"**,  
 **"Прищепа Екатерина Михайловна"**,  
 **"Артемьева Дарья Сергеевна"**,  
 **"Попова Софья Александровна"**,  
 **"Башмакова Анастасия Алексеевна"**,  
 **"Корнева Татьяна Андреевна"**,  
 **"Олзошкина Янжина Владленовна"**,  
 **"Касьянов Максим Евгеньевич"**,  
 **"Олейник Анастасия Александровна"**,  
 **"Сивухов Артём Олегович"**,  
 **"Груздев Всеволод Алексеевич"**,  
 **"Буковец Данила Андреевич"**,  
 **"Зелянина Алёна Геннадьевна"**,  
 **"Мерзляков Данила Артемович"**,  
 **"Карасёв Артём Владимирович"**,  
 **"Пономаренко Александр Павлович"**,  
 **"Курносиков Кирилл Андреевич"**,  
 **"Гуриков Дмитрий Олегович"**,  
 **"Котова Екатерина Дмитриевна"**,  
 **"Лихачев Марк Игоревич"**,  
 **"Волкова Татьяна Алексеевна"**,  
 **"Марунько Анна Сергеевна"**,  
 **"Пашкевич Денис Вячеславович"**,  
 **"Маркова Ольга Алексеевна"**,  
 **"Термышева Полина Евгеньевна"**,  
 **"Василевская Лидия Игоревна"**,  
]  
  
  
**class** Task1():  
 *"""  
 Создать словарь адресной книги, содержащий ФИО и адрес. Заполнить его 50 элементами, реализовать поиск по адресу  
 """* **def** \_\_init\_\_(self):  
 self.dict\_generator()  
 self.search()  
  
 **def** dict\_generator(self):  
 n = UNIC\_NAMES\_LIST  
 self.d = {}  
 **for** i **in** range(0, len(n), 2):  
 self.d[**"адрес №"** + str(i)] = [n[i], n[i + 1]]  
  
 **def** search(self):  
 s = input(**"Введите адрес для поиска ->"**)  
 **if** s **in** self.d:  
 print(**"Адрес найден!\nПроживающие по адресу:"**)  
 [print(x) **for** x **in** self.d[s]]  
 **else**:  
 **for** k, v **in** self.d.items():  
 **for** p **in** v:  
 **if** s **in** p:  
 print(**"Возможно вы имели ввиду '"** + p + **"' по '"** + k + **"'"**)  
  
  
**class** Task2():  
 *"""  
 Cоздать словарь телефонного справочника. Заполнить его 50 элементами. Реализовать поиск по телефону  
 """* **def** \_\_init\_\_(self):  
 self.dict\_generator()  
 self.search()  
  
 **def** locale\_random(self, n):  
 **return** str(randint(10 \*\* (n - 1), (10 \*\* n) - 1))  
  
 **def** dict\_generator(self):  
 self.d = {}  
 **for** e **in** UNIC\_NAMES\_LIST:  
 key = **"+7"** + self.locale\_random(10)  
 self.d[key] = e  
 print(**"Сгенерировали ключ "** + key)  
  
 **def** search(self):  
 input\_key = input(**"Введите номер телефона для поиска ->"**)  
 **if** input\_key **in** self.d:  
 print(**"Значение для телефона "** + input\_key + **" -> "** + self.d[input\_key])  
 **else**:  
 print(**"Введённого номера телефона нет в базе"**)  
  
  
**class** Task3(object):  
 *"""  
 реализовать проверку на существующие записи в предыдущих заданиях с возможностью дополнения  
 """* **def** \_\_init\_\_(self):  
 d = {  
 **"1"**: Task1Plus,  
 **"2"**: Task2Plus,  
 }  
 e = input(**"Какой номер вы хотите дополнить ?"**)  
 **if** e **in** d:  
 d[e]()  
 **else**:  
 print(**"Такого номера не существует!"**)  
  
  
**class** Task1Plus(Task1):  
 *"""  
 Класс для дополнения задания №1  
 """* **def** \_\_init\_\_(self):  
 self.dict\_generator()  
 self.search()  
  
 **def** search(self):  
 s = input(**"Введите адрес для поиска ->"**)  
 **if** s **in** self.d:  
 print(**"Адрес найден!\nПроживающие по адресу:"**)  
 [print(x) **for** x **in** self.d[s]]  
 **else**:  
 print(**"Адрес не найден, но мы его добавим в систему"**)  
 names = input(**"Введите ФИО людей, проживающих по этому адресу через заптую -> "**)  
 self.d[s] = names.split(**","**)  
 print(**"Обновлённый словарь:"**)  
 **for** k, v **in** self.d.items():  
 print(k, v)  
  
  
**class** Task2Plus(Task2):  
 *"""  
 Класс для дополнения задания №2  
 """* **def** \_\_init\_\_(self):  
 self.dict\_generator()  
 self.search()  
  
 **def** search(self):  
 s = input(**"Введите телефон для поиска ->"**)  
 **if** s **in** self.d:  
 print(**"Телефон найден!\nАбонент "** + self.d[s])  
 **else**:  
 print(**"Абонент не найден, но мы его добавим в систему"**)  
 name = input(**"Введите ФИО абонента -> "**)  
 self.d[s] = name  
  
 print(**"Обновлённый словарь:"**)  
 **for** k, v **in** self.d.items():  
 print(k, v)  
  
  
**class** Task4(object):  
 *"""  
 создать словарь на свободную тему, включающий в себя кортеж в качестве ключа, реализовать поиск  
 """* **def** \_\_init\_\_(self):  
 self.d = {}  
 self.max\_values = 100  
 self.hashtable\_generator()  
 self.results\_searcher()  
  
 **def** all\_counter(self, \*t):  
 res = 0  
 **for** i **in** t:  
 res += i  
 **return** res  
  
 **def** all\_multipy(self, \*t):  
 res = 1  
 **for** i **in** t:  
 res \*= i  
 **return** res  
  
 **def** hashtable\_generator(self):  
 **for** i **in** range(self.max\_values):  
 **for** j **in** range(self.max\_values):  
 **for** k **in** range(self.max\_values):  
 self.d[(i, j, k)] = [  
 {**"multiplication"**: self.all\_multipy(i, j, k), **"sum"**: self.all\_counter(i, j, k)}]  
  
 **def** check\_digit(self, e):  
 **try**:  
 **return** int(e)  
 **except**:  
 **return** e  
  
 **def** results\_searcher(self):  
 out\_str = **"Введите 3 числа через пробел от 0 до 100 для быстрого подсчёта их суммы и произведения -> "** r = self.d[tuple([self.check\_digit(x) **for** x **in** input(out\_str).split(**" "**)])][0]  
 print(**"Произведение: "** + str(r[**"multiplication"**]) + **"\nСумма: "** + str(r[**"sum"**]))  
  
  
**class** Task5(object):  
 *"""  
 создать словарь авиарейсов, в возможностью поиска маршрута из точки А в точку В с учётом 1 пересадки  
 """* **def** \_\_init\_\_(self):  
 self.d = {  
 **"Москва"**: [**"Лондон"**, **"Владивосток"**, **"Санкт-Петербург"**],  
 **"Лондон"**: [**"Москва"**, **"Сингапур"**],  
 **"Сингапур"**: [**"Лондон"**],  
 **"Калининград"**: [**"Санкт-Петербург"**],  
 **"Санкт-Петербург"**: [**"Калининград"**, **"Москва"**],  
 **"Владивосток"**: [**"Москва"**, **"Норильск"**],  
 **"Норильск"**: [**"Владивосток"**],  
 }  
  
 self.search()  
  
 **def** search(self):  
  
 search\_flag = **False** point\_a = input(**"Введите точку А -> "**)  
 point\_b = input(**"Введите точку В -> "**)  
  
 **if** point\_a **in** self.d:  
 **for** tranzit\_city **in** self.d[point\_a]:  
 **if** tranzit\_city == point\_b:  
 search\_flag = **True** print(**"\*План перелёта\*\n"** + point\_a + **" -> "** + point\_b)  
 **break  
  
 for** city **in** self.d[tranzit\_city]:  
 **if** city == point\_b:  
 search\_flag = **True** print(  
 **"\*План перелёта\*\n"** + point\_a + **" -> "** + tranzit\_city + **"\n"** + tranzit\_city + **" -> "** + city)  
  
 **if** search\_flag == **False**:  
 print(**"По вашему запросу ничего не найдено"**)  
  
 **else**:  
 print(**"Нет точки А в исходном словаре"**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 d = {  
 **"1"**: Task1,  
 **"2"**: Task2,  
 **"3"**: Task3,  
 **"4"**: Task4,  
 **"5"**: Task5  
 }  
 s = input(**"Введите номер задания -> "**)  
 **if** s **in** d:  
 d[s]()  
 **else**:  
 print(**"Такого задания нет!"**)

**class** Task6():  
 *"""  
 Создать словарь железнодорожных сообщений с учетом более одной но менее 4 пересадок, с рекомендацией оптимального маршрута по времени  
 Между переходами разница в 30 мин  
 """* **def** \_\_init\_\_(self):  
 self.TIME\_WAIT = 30  
 self.all\_ways\_list = []  
 self.d = {  
 **"Одинцово"**: [  
 {**"time"**: 30, **"name"**: **"Белорусский вокзал"**},  
 {**"time"**: 3, **"name"**: **"Баковка"**},  
 {**"time"**: 15, **"name"**: **"Отрадное"**},  
 ],  
  
 **"Баковка"**: [  
 {**"time"**: 3, **"name"**: **"Одинцово"**},  
 {**"time"**: 10, **"name"**: **"Курский Вокзал"**},  
 {**"time"**: 25, **"name"**: **"Савёловский Вокзал"**},  
 ],  
 **"Отрадное"**: [  
 {**"time"**: 15, **"name"**: **"Одинцово"**},  
 {**"time"**: 60, **"name"**: **"Курский Вокзал"**},  
 {**"time"**: 38, **"name"**: **"Савёловский вокзал"**},  
 ],  
 **"Белорусский вокзал"**: [  
 {**"time"**: 30, **"name"**: **"Одинцово"**},  
 {**"time"**: 10, **"name"**: **"Курский Вокзал"**},  
 {**"time"**: 5, **"name"**: **"Савёловский вокзал"**},  
 ],  
 **"Курский Вокзал"**: [  
 {**"time"**: 10, **"name"**: **"Белорусский Вокзал"**},  
 {**"time"**: 10, **"name"**: **"Баковка"**},  
 {**"time"**: 60, **"name"**: **"Отрадное"**},  
 ],  
 **"Савёловский Вокзал"**: [  
 {**"time"**: 5, **"name"**: **"Белорусский вокзал"**},  
 {**"time"**: 25, **"name"**: **"Баковка"**},  
 {**"time"**: 38, **"name"**: **"Отрадное"**},  
 ],  
 }  
 self.way\_inputer()  
 self.way\_recognizer()  
 **if** self.all\_ways\_list != []:  
 self.time\_detector()  
 self.result\_outputer()  
  
 **def** result\_outputer(self):  
 self.all\_ways\_list.sort(key=**lambda** x: (x[**"times"**]))  
 **for** e **in** self.all\_ways\_list:  
 buf\_out = **""  
 for** way **in** e[**"points"**]:  
 buf\_out += way + **" -> "** buf\_out = buf\_out[:len(buf\_out) - 4]  
 print(**"\*"** \* 40 + **"\nПуть:\n"** + buf\_out)  
 print(**"Время: "** + str(e[**"times"**]) + **" минут"**)  
  
 **def** time\_detector(self):  
  
 all\_ways\_list = self.all\_ways\_list  
  
 **for** i **in** range(len(all\_ways\_list)):  
 total\_time = 0  
 **if** len(all\_ways\_list[i][**"points"**]) > 2:  
 *# Цикл по каждому времени* **for** time **in** all\_ways\_list[i][**"times"**]:  
 total\_time += time + self.TIME\_WAIT  
 total\_time -= self.TIME\_WAIT  
 **else**:  
 total\_time = all\_ways\_list[i][**"times"**][0]  
  
 all\_ways\_list[i][**"times"**] = total\_time  
  
 self.all\_ways\_list = all\_ways\_list  
  
 **def** way\_inputer(self):  
  
 self.point\_a = input(**"Введите точку А -> "**)  
 self.point\_b = input(**"Введите точку В -> "**)  
  
 **def** way\_recognizer(self):  
  
 d = self.d  
 point\_a = self.point\_a  
 point\_b = self.point\_b  
  
 all\_ways\_list = []  
  
 **if** point\_a **in** d:  
  
 **for** first\_element **in** d[point\_a]:  
  
 **if** first\_element[**"name"**] == point\_b:  
 print(**"[Вложенность 0] "** + point\_a + **" - > "** + first\_element[**"name"**])  
 all\_ways\_list.append(  
 {  
 **"points"**: [point\_a, first\_element[**"name"**]],  
 **"times"**: [first\_element[**"time"**]],  
 **"detector\_number"**: 0,  
 })  
  
 **if** first\_element[**"name"**] **in** d:  
 **for** second\_element **in** d[first\_element[**"name"**]]:  
 **if** second\_element[**"name"**] == point\_b:  
 print(**"[Вложенность 1] "** + point\_a + **" - > "** + first\_element[**"name"**] + **" -> "** +  
 second\_element[**"name"**])  
 all\_ways\_list.append(  
 {  
 **"points"**: [point\_a, first\_element[**"name"**], second\_element[**"name"**]],  
 **"times"**: [first\_element[**"time"**], second\_element[**"time"**]],  
 **"detector\_number"**: 1,  
 })  
  
 **if** second\_element[**"name"**] **in** d:  
 **for** third\_element **in** d[second\_element[**"name"**]]:  
 **if** third\_element[**"name"**] == point\_b:  
 print(**"[Вложенность 2] "** + point\_a + **" - > "** + first\_element[**"name"**] + **" -> "** +  
 second\_element[**"name"**] + **" -> "** + third\_element[**"name"**])  
 all\_ways\_list.append(  
 {  
 **"points"**: [point\_a, first\_element[**"name"**], second\_element[**"name"**],  
 third\_element[**"name"**]],  
 **"times"**: [first\_element[**"time"**], second\_element[**"time"**],  
 third\_element[**"time"**]],  
 **"detector\_number"**: 2,  
 })  
  
 **if** third\_element[**"name"**] **in** d:  
 **for** fourth\_element **in** d[third\_element[**"name"**]]:  
 **if** fourth\_element[**"name"**] == point\_b:  
 print(**"[Вложенность 3] "** + point\_a + **" - > "** + first\_element[  
 **"name"**] + **" -> "** + second\_element[**"name"**] + **" -> "** + third\_element[  
 **"name"**] + **" -> "** + fourth\_element[**"name"**])  
 all\_ways\_list.append(  
 {  
 **"points"**: [point\_a, first\_element[**"name"**], second\_element[**"name"**],  
 third\_element[**"name"**], fourth\_element[**"name"**]],  
 **"times"**: [first\_element[**"time"**], second\_element[**"time"**],  
 third\_element[**"time"**], fourth\_element[**"time"**]],  
 **"detector\_number"**: 3,  
 })  
  
 self.all\_ways\_list = all\_ways\_list  
  
 **else**:  
 print(**"Нет исходной точки в начале"**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 Task6()