A dictionary of stations is given, where the keys are the names of Bicikelj stations and the values are dictionaries with the geographical coordinates of the stops, capacity (number of stands), and the current number of bikes at the station. (See the dictionary in the tests.)

The provided function **`razdalja_geo(lat1, lon1, lat2, lon2)`** returns the distance in kilometers between two points for a given pair of geographical coordinates. Use at will. (The function works correctly within a city region.)

### 0. Distances

Write a function **`razdalja(postaja1, postaja2)`** that takes the names of two stations and returns the distance between them. (The function is worth 0 points. You write it for warm-up and for use in the following tasks.)

### 1. Closest 3

The author of the exam is not a Bicikelj user (once was enough), but knows that at a certain station you sometimes cannot get or return a bike, so it is necessary to look for nearby stations. Write a function **`najblizje3(postaja)`** that, for a given station, finds the three closest stations and returns a list of (three) pairs (distance, name). The list should be sorted by distance.

The call **`najblizje3("ŽIVALSKI VRT")`** returns **`[(0.73687236710574, 'VIŠKO POLJE'), (0.9639016462683163, 'TEHNOLOŠKI PARK'), (1.0760703913614853, 'SOSESKA NOVO BRDO')]`**.

### 2. Angelca's Riders

Angelca from the Department of Forestry Activities and Motor Traffic likes historical movies and knows that in the past there were stations where, after a certain distance, horses were changed. Since Bicikelj, according to Angelca, is the same, they will now limit the length of the trip allowed with a rented bike: the distance between the station where the bike is rented and the station where it is returned must not exceed a prescribed distance.

For longer distances, it will be necessary to ride from station to station and change bikes: someone will rent a bike at Cankarjeva ulica, ride it to Bavarski dvor, return it there and take the next one to Vilharjeva cesta … and so on. Angelca says this will be really cool.

If the allowed distance is too short, the map will break into "islands" of mutually reachable stations – that is, we shall have groups of stations such that no other station is within the prescribed distance from any of them.

Write a function **`dosegljive(postaja, razdalja, otok)`** that takes the name of a station, the maximum allowed distance, and an empty set **`otok`**. The function does not return anything, but must store into the given set otok all the stations that, under the given distance restriction, are directly or indirectly (through other stations) reachable from the starting station (including the starting station).

```
otok = set()
dosegljive("TEHNOLOŠKI PARK", 0.7, otok)
print(otok)
```

prints `{'VIŠKO POLJE', 'SOSESKA NOVO BRDO', 'TEHNOLOŠKI PARK'}`.

### 3. Isolated Station

Write a function **`izolirana()`** that finds and returns the name of the station that is farthest from its nearest station. For these specific stations, this is Zalog, but the function must be written in a general way.

### 4. Write and Read

Write a function **`zapisi(ime_datoteke)`** that gets a name of a file into which it writes the station names (padded to 40 characters), followed by the number of available bikes and the number of stands. The output must be sorted alphabetically by station names (characters with diacritics – č, š, ž – will come last; use the default Python sorting) and in the format shown in the box.

```
KONGRESNI TRG-ŠUBIČEVA ULICA              9/20
KOPALIŠČE ILIRIJA                         0/20
KOPALIŠČE KOLEZIJA                       19/20
KOPRSKA ULICA                             3/8
KOSEŠKI BAJER                            14/20
```

Write a function **`preberi(ime_datoteke)`** that reads such a file into a dictionary. The keys of the dictionary are the station names, and the corresponding values are pairs with the number of bikes and the number of stands.

### 5. Coordinator

Write a class Koordinator.

- The constructor takes no arguments. It may store whatever you want. For example, data from postaje. Or not.
- `stanje(postaja)` returns the number of available bikes at the given station.
- `odpelji(postaja)` returns `True` if there is a bike at the station and `False` if not. If there is a bike, it remembers that there is one bike fewer at the station.
- `vrni(postaja)` returns `True` if there is a free stand for a bike at the station and False if not. If there is space, it remembers that there is one more bike at the station.

Important: the class must not modify the values in the (global) dictionary postaje!