Železniške povezave so podane v obliki slovarja. Ključi so imena postaj (npr. Zidani most), pripadajoče vrednosti pa seznami povezav na neposredno povezane postaje. Povezava je trojka z imenom ciljne postaje (npr. Laško), časom odhoda (v minutah od polnoči) s postaje v ključu (Zidani most) in časom prihoda na ciljno postajo (npr. Laško). Če ključu "Zidani Most" pripada seznam `[('Laško', 364, 382), ('Hrastnik', 365, 386), ('Laško', 375, 390), ('Radeče', 379, 395)]`, iz Zidanega Mostu odhaja vlak v Laško ob 6:04 in prispe tja ob 6:22, nato v Hrastnik ob 6:05 s prihodom ob 6:26, potem spet v Laško ob 6:15 s prihodom ob 6:30, potem v Radeče ob 6:19 s prihodom ob 6:35. **Povezave so urejene po časih odhodov.** Časi so izmišljeni in vsaka povezava z resničnimi prihodi in odhodi vlakov je zgolj naključna. V tem smislu so podobni dejanskim voznim redom Slovenskih železnic.

1. Postaje

Napišite funkcijo **`postaje(povezave)`**, ki prejme slovar s povezavami in vrne tri množice: množico *končnih* in *prehodnih* postaj ter množico *križišč*. Postaja je *končna*, če vlaki z nje vozijo na eno samo postajo, *prehodna*, če na dve in *križišče*, če vlaki s te postaje vozijo na več kot dve postaji. Vse povezave so dvosmerne: če vozi vlak iz A v B, vozi tudi iz B v A.

2. Potovalni časi

Napišite funkcijo **`naslednja_povezava(povezave, odkod, kam, cas)`**, ki vrne par s časom, ko odide naslednji vlak ob podanem času `cas` (ali po njem) iz `odkod` v `kam`, in časom prihoda v kam. Klic `naslednja_povezava(povezave, "Zidani Most", "Laško", 370)` vrne `(375, 390)`. Če vlaka po podanem času ni, vrne `None`.

Napišite funkcijo **`potovalni_cas(povezave, pot, zacetek)`**, ki vrne čas (v minutah), ki ga bo potnik potreboval, da prepotuje podano `pot` (seznam imen postaj), če se nanjo odpravi ob času `zacetek`. Potnik na vsaki postaji uporabi prvi možni vlak do naslednje postaje. Če poti ni možno opraviti, ker določena povezava ne obstaja ali pa na tisti dan ni več vlaka na tej povezavi, funkcije vrne `None`. Brez skrbi smete predpostaviti, da vlaki Slovenskih železnic nimajo zamud.

Klic `potovalni_cas(povezave, ["Kranj", "Škofja Loka", "Ljubljana", "Litija", "Hrastnik"], 420)` vrne 165. Potnik ob 439 sede na vlak Kranj – Škofja Loka, kamor prispe ob 464 in takoj nadaljuje v Ljubljano s prihodom ob 480. Do 542 čaka na vlak, ki ga ob 565 pripelje v Litijo, takoj nadaljuje v Hrastnik in prispe ob 585. 585 – 420 = 165.

3. Vozni redi

Vozni redi so v resnici sestavljeni tako, da posamični vlak prevozi določeno relacijo, na primer Ljubljana – Škofja Loka – Kranj – Jesenice. Vsak vlak, ki gre iz Ljubljane v Škofjo Loko, že ob isti uri nadaljuje pot v Kranj in nato do Jesenic: če nek vlak prispe ob 800 iz Ljubljana v Škofjo Loko, odide nek vlak ob 800 iz Škofje Loke v Kranj.

Napišite funkcijo **`vozni_red(povezave, linija, ime_datoteke)`**, ki prejme slovar povezav, seznam `linija`, ki vsebuje imena krajev (npr `["Ljubljana", "Škofja Loka", "Kranj", "Jesenice"]`) in ime datoteke. V datoteko mora izpisati vozni red, kot ga kaže okvirček. Imena postaj so izpisana na 20 mest in pred vsakim časom sta dva presledka. (Za razmislek: druga vrstica, recimo, pravzaprav vsebuje ravno ure vseh odhodov iz Škofje Loke v Kranj, ki so hkrati tudi ure vseh prihodov iz Ljubljane v Škofjo Loko. Prva vrstica pa seveda vsebuje le (vse) odhode in zadnja vse prihode.)

```
           Ljubljana  05:59  06:58  07:57  08:53  09:53  10:57  11:54  12:54  13:55  14:59  15:54
         Škofja Loka  06:18  07:15  08:21  09:08  10:16  11:20  12:09  13:14  14:11  15:15  16:16
              Kranj  06:36  07:31  08:42  09:24  10:34  11:41  12:26  13:33  14:27  15:35  16:39
            Jesenice  06:54  07:56  08:57  09:42  10:59  11:59  12:47  13:50  14:48  15:55  16:55
```

4. Izračun poti

Napišite funkcijo **`cas_prihoda(povezave, odkod, kam, zacetek, omejitev)`**, ki prejme slovar, začetno in končno postajo, čas, ko se bo nekdo odpravil na pot s postaje `odkod` in skrajni čas, do katerega mora prispeti. Funkcija mora vrniti najzgodnejši čas, ko lahko pride do `kam`, ob predpostavki, da poišče optimalno zaporedje postaj, da mu prestopanja ne vzamejo časa in da vlaki ne zamujajo. :) Primer najdete v testih.

Pomemben spoiler: testi sicer nastavijo omejitev na neko veliko število. Ko vaša funkcija že najde neko možno pot, pa naj jo v rekurzivnem klicu zaostri, da znotraj njih ne bo iskala poti, ki bi vzele več časa kot najkrajša pot, ki ste jo našli doslej. Brez tega bo funkcija prepočasna.

5. Potnik

Napišite razred **`Potnik`**. Razred naj predpostavi, da se povezave nahajajo v globalni spremenljivki z imenom `povezave`.

- Konstruktor prejme začetno postajo in trenutni čas.
- Metoda **`premik(kam)`** prejme postajo, na katero potnik potuje s postaje, na kateri se trenutno nahaja. Premik opravi v najzgodnejšem možnem času. Če določenega premika ni možno opraviti (ker povezava ne obstaja ali pa je potnik zamudil zadnjo povezavo v podanem dnevu), funkcija ne naredi ničesar.
- Metoda **`kje()`** vrne postajo, na kateri se trenutno nahaja potnik, in čas, ko je prispel nanjo.
- Metoda **`izguba()`** vrne skupni čas čakanja na povezave (ne skupni čas potovanja!).