1. V karanteno

Aktivnosti oseb shranimo v slovar: `{"Ana": ["kava", "trgovina", "burek"], "Berta": ["telovadba", "frizer"], "Ema": ["kava", "telovadba"], "Fanči": ["frizer"], "Greta": ["lokostrelstvo", "curling"]}`. Napiši funkcijo `v_karanteno(aktivnosti, okuzene)`, ki prejme slovar, kakršen je gornji in seznam okuženih oseb. Vrne naj množico oseb, ki so se udeležile katere od aktivnosti, ki se jih je udeležila katera od okuženih oseb. Klic `v_karanteno(aktivnosti, ["Ema", "Berta"])` vrne `{"Ana", "Berta", "Ema", "Fanči"}`; Ano vrne, ker sta bili obe z Emo na kavi, Fanči pa, ker sta bili z Berto pri frizerju. V množici morajo biti tudi vse osebe, ki so v podanem seznamu okuženih.

2. Genom SARS-Covid-19

Napiši funkcijo `trojke(s, n)`, ki prejme zaporedje baz v mRNA in vrne `n` najpogostejših zaporednih trojk, urejenih po pogostosti. Če sta dve trojki enako pogosti, ima prednost tista, ki je kasneje po abecedi (ker je tako lažje sprogramirati!). Če je različnih trojk manj kot `n`, jih vrne pač, kolikor jih je. V zaporedju `"acgtacgatacgacg"` je najpogostejša trojka acg (štirikrat), sledijo tac in cga (dvakrat), nato gta, gat, gac, cgt in ata (enkrat). Klic `trojke("acgtacgatacgacg", 5)` zato vrne `["acg", "tac", "cga", "gta", "gat"]`.

3. Statistika

Število okuženih v zadnjih dneh po posameznih državah je podano v nizih v naslednji obliki:

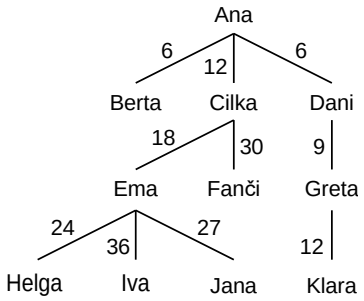
```
Slovenija:31,20,25,14,50,60
Hrvaška:150,170,200,220,221
Madžarska:100,70,35
```

Napiši funkcijo `statistika(podatki, drzava, n)`, ki prejme takšen niz ter vrne število okuženih v zadnjih `n` dneh v podani državi. Za manjkajoče podatke predpostavi, da je bilo takrat okuženih 0 oseb. `statistika(podatki, "Slovenija", 3)` za gornje podatke vrne, 124 (to je 14 + 50 + 60); `statistika(podatki, "Madžarska", 5)` vrne 205.

4. Okuženi

Ana je bila okužena na dan 0. Na dan 6 je okužila Berto in Dani, na dan 12 pa Cilko. Berta ni okužila nikogar. Cilka je na dan 18 okužila Emo in na dan 30 Fanči. In tako naprej.

```
okuzbe = {"Ana": {"Berta": 6, "Cilka": 12, "Dani": 6},
          "Berta": {},
          "Cilka": {"Ema": 18, "Fanči": 30},
          "Dani": {"Greta": 9},
          "Ema": {"Helga": 24, "Iva": 36, "Jana": 27},
          "Fanči": {},
          "Greta": {"Klara": 12},
          "Helga": {},
          "Iva": {},
          "Jana": {},
          "Klara": {}}
```



Slediti želimo okužbam, ki izhajajo iz določene osebe. Napiši funkcijo `okuzeni(dan, oseba, okuzbe)`, ki vrne množico oseb (vključno s podano osebo), ki so prek podane osebe okužene do (vključno) podanega dneva. Klic `okuzeni(26, "Cilka", okuzbe)` vrne `{"Cilka", "Ema", "Helga"}`; prek Cilke se okužita tudi Fanči in Jana, vendar šele po dnevu 26.

5. Sledilnik

Prva naloga je seveda naivna: pomemben je tudi čas aktivnosti in ne le aktivnost sama (kraj pa bomo zanemarili). Napiši razred `Oseba`, ki bo hranil aktivnosti določene osebe. Razred naj ima konstruktor brez argumentov in metode:

- konstruktor brez argumentov;
- `aktivnost(kaj, kdaj)` zabeleži, da je oseba ob podanem času opravljala podano aktivnost;
- `vse_aktivnosti()` vrne množico vseh aktivnosti, s katerimi se je ukvarjala oseba.
- `mozna_okuzba(druga_oseba)` vrne `True`, če sta ta oseba in podana druga_oseba kdaj ob istem času opravljali isto aktivnost (npr. istočasno pili kavo), sicer pa `False`.

Iz razreda `Oseba` izpelji razred `VarnaOseba`, ki ima drugačen konstruktor in metodo `mozna_okuzba`. Konstruktor prejme množico aktivnosti, pri katerih podana oseba nosi masko. Metoda `mozna_okuzba(druga_oseba)` zdaj vrne `True`, če sta osebi kdaj ob istem času opravljala isto aktivnost, pri kateri `druga_oseba` (ki je prav tako objekt tipa `VarnaOseba`) ni nosila maske. Če jo je nosila ta oseba (`self`), to ne pomaga, ker maska ščiti druge in ne tistega, ki jo nosi.