Programiranje 2:
Koncepti programskih jezikov

3. domača naloga

25. maj 2021

Naloge se rešuje samostojno, v primeru prepisovanja se upošteva univerzitetni pravilnik. Oddajte eno TXT datoteko z OCaml kodo na `http://e.famnit.upr.si`. Pred oddajo nujno testirajte svojo kodo, ki se mora pravilno prevesti[1]. Če oddate kodo, ki se ne prevede, bo vaš izdelek točkovan z 0 točkami. Rešitev naj vsebuje vaše ime, priimek, smer in vpisno številko – ti podatki naj bodo vsebovani kot komentar. Poskusite rešiti vsako oštevilčeno nalogo kot eno samo funkcijo, razred ali metodo.

Srečno!

1 Potovalna Agencija 60%

Spodaj je podana definicija tipa experience za potrebe te naloge:

```
type experience = Beginner | Intermediate | Experienced | WorldClass | AI
```

1. Napiši razred *driver*, ki hrani vrednost tipa experience, ki predstavlja izkušenost voznika avtobusa. Razred tudi hrani vrednost (float) za maksimalno razdaljo, ki jo voznik lahko prevozi. Napiši getter, setter in toString metode. (2%)

2. Napiši razred *bus*, ki hrani vrednost za trenutne proste sedeže (int), doseg (float), najemnino (float) v enoti EUR/potnika, porabo goriva na km (float) in število sedežev (int). Bus je prvotno prazen zato se vrednost za trenutno proste sedeže ujema s številom sedežev. Napiši getter, setter in toString metode.(4%)

   (a) Napiši metodo *addPassenger*, ki ima za argument število potnikov in vrne true, če jih je manj ali enako kot prostih sedežev (ter popravi število prostih sedežev), in false v nasprotnem primeru (in ne spremeni števila prostih sedežev, saj želi skupina potovati skupaj).(4%)

3. Napiši razred *travelAgency*, ki vsebuje seznam avtobusov (bus list), maximalno število avtobusov ki, jih lahko hranimo (int) in seznam šoferjev (driver list). Polega tega hrani tudi (bus*driver) list. Napiši getter, setter in toString metode. (7%)

   (a) Napiši metodo *addbus*, ki ima argument tipa bus, in ga doda na seznam avtobusov, pod pogojem da je dolžina seznama avtobusov manjša od maximalnega števila avtobusov . Poleg tega metoda vrne true, če je avtobus uspešno dodan, in vrne false drugače. (4%)

   (b) Napiši metodo *addDriver*, ki ima argument tipa driver, in ga doda na seznam šoferjev. (4%)

   (c) Napiši metodo *randomSchedule*, ki vsakemu avtobusu dodeli naključnega šoferja (šoferji se lahko ponovijo). Funkcija naj vrne unit (), rezultat pa naj shrani v polje s tipom (bus*driver) list. (10 %)

   Modul Random vsebuje funkcije za delo z naključnimi števili. Slednji primer kaže kako premešamo seznam.

[1] Uspešno prevajanje lahko preverite s pomočjo orodja dosegljivega na `https://try.ocamlpro.com/`.

```
let shuffle d =
   let nd = List.map (fun c -> (Random.bits (), c)) d in
      let sond = List.sort compare nd in
         List.map snd sond;;
shuffle [1;2;3;4];;
```

(d) Napiši metodo *rentBus*, ki ima za argument število potnikov in razdaljo . Metoda najprej požene metodo randomSchedule, in nato izmed vrnjenih naključnih (bus,driver) parov, prefiltrira tiste, pri kateri voznik ne more opraviti tako dolge poti, avtobus ne more opraviti tako dolge pot ali avtobus nima dovolj prostih mest. Nato iz prefiltriranega seznama parov izbere najcenejši par. Cena je glede na ceno voznika + ceno goriva + ceno najema avtobusa*(št. potnikov v argumentu). Vse razdalje se smatrajo, kot da so v km. (10%)

Cene šoferjev so sledeče:

- Beginner: 1 EUR / km
- Intermediate: 2 EUR / km
- Experienced: 3 EUR / km
- World Class: 5 EUR / km
- AI: 20 EUR / potovanje

(e) Napiši medodo *optimizedRent*, ki ima za argument število potnikov in razdaljo. Metoda najde najcenejši avtobus (iz seznama avtobusov), ki ima dovolj prostih sedežev in lahko opravi razdaljo in najcenejšega voznika (iz seznama voznikov) sposobnega za tako dolgo pot. Metoda vrne šoferja in avtobus kot urejen par. Cena je glede na ceno voznika + ceno goriva + ceno najema avtobusa*(št. potnikov v argumentu). Vse razdalje se smatrajo, kot da so v km. (15%)

2 Nalezljiva vozlišča 40%

Definiran je naslednji tip:

```
type graph_term = {nodes : char list;  edges : (char * float * char) list};;
```

Naj bo $G = (V, A)$ usmerjen graf (oz. omrežje), kjer imamo množico vozlišč V v seznamu poimenovanem `nodes`, množica usmerjenih povezav $A \subseteq V \times V$ pa je shranjena pod imenom `edges`, ter je podana kot seznam trojic elementov tipa `(char * float * char)`. Če seznam usmerjenih povezav vsebuje element (v_1, w, v_2), potem obstaja usmerjena povezava $v_1 \to v_2$ z utežjo w.[2]

1. Napiši funkcijo `oneStepInfection` ki sprejme dva parametra: usmerjen graf, predstavljen s pomočjo zgoraj definiranega tipa, ter eno vozlišče, podano z imenom (`char`). Proces okužbe se zgodi na podlagi uteži in v korakih. Okuženo vozlišče v bo v enem koraku okužilo vsa vozlišča u, za katera obstaja povezava (v, w, u), pri čemer mora veljati $w > 0,6$. Funkcija naj simulira en korak take okužbe, ter vrne seznam okuženih vozlišč. (20 %)

2. Napišite funkcijo `infectionProcess`, ki kot vhod vzame enake parametre kot `oneStepInfection`: usmerjen graf, predstavljen s pomočjo zgoraj definiranega tipa, ter eno vozlišče, podano z imenom (`char`). Funkcija naj vrne seznam vseh vozlišč v grafu, ki postanejo okužena po enem ali več korakov.[3] (10%)

3. Napiši funkcijo `mostInfectious` ki kot vhod vzame usmerjen graf, predstavljen s pomočjo zgoraj definiranega tipa ter vrne vozlišče ki okuži največ drugih vozlišč, glede na prej definirano funkcijo `infectionProcess`. (10%)

[2] Za morebitne dodatne osnovne informacije o usmerjenih grafih glej `https://en.wikipedia.org/wiki/Glossary_of_graph_theory`.

[3] Vozlišča ne umrejo, ter se tudi ne morejo ozdraviti.

```
# type graph_term = {nodes : char list;  edges : (char * float * char) list};;
# let samplegraph:graph_term={nodes= ['1';'2';'3';'4';'5'];edges=[('1',0.3,'2');
('2',0.7,'1');('1',10.1,'3');('3',3.,'4');('3',0.1,'5');('4',7.,'5');('2',6.,'3')]};;
# oneStepInfection samplegraph '1';;
- : char list = ['1'; '3']
# oneStepInfection samplegraph '2';;
- : char list = ['2'; '1'; '3']
# oneStepInfection samplegraph '5';;
- : char list = ['5']
# infectionProcess samplegraph '1';;
- : char list = ['1'; '3'; '4'; '5']
# infectionProcess samplegraph '2';;
- : char list = ['2'; '1'; '3'; '4'; '5']
# mostInfectious samplegraph;;
- : char = '2'
```