RP3 2021/2022

1. zadaća

Rok predaje: 23. 12. 2021. u 23:59 (Merlin)

U prvom dijelu zadaće definiramo jedan deklarativan "jezik" nalik JSON-u, a u drugom dijelu zadaće iskorištavamo taj jezik za specifikaciju (vrlo jednostavnih) sučelja. U zadaći se ne boduje efikasnost (složenost). Svi rubni slučajevi koji nisu pokriveni tekstom zadaće ostavljeni su vama na izbor. Prije implementacije svakako pogledajte primjere na kraju zadaće.

1. dio (5 bodova)

U ovom dijelu zadaće definiramo tip (klasu ili strukturu, po izboru) `node`. Instanca ovog tipa predstavlja jedan među sljedećim objektima:

- cijeli broj;
- tekst sastavljen od nula ili više sljedećih simbola: slova engleske abecede (a–z, A–Z), znamenke (0–9) i praznina (`' '`);
- konačan (možda prazan) niz proizvoljne duljine, čiji su svi elementi instance tipa `node`;
- konačan (možda prazan) neuređeni rječnik (*map*, *dictionary*) proizvoljne duljine, koji pojedinom stringu pridružuje instancu tipa `node`.

Klasa `node` mora imati konstruktor ili konstruktore koji može primiti (barem) bilo koji od sljedećih tipova, koji redom odgovaraju ranije navedenom mogućem sadržaju pojedine instance tipa `node`:

- `int`;
- `string`;
- `List<node>`;
- `Dictionary<string, node>`.

Konstruktor mora nekako pohraniti tako primljeni objekt (sami odaberite implementacijske detalje). Ne morate provjeravati odgovara li proslijeđeni objekt specifikaciji (npr. u slučaju da se radilo o stringu, sadrži li samo dopuštene simbole). Možete pretpostaviti da nakon konstrukcije neće biti nikakvih pokušaja promjene sadržaja.

Primjer konstrukcije složenijeg objekta:

```
new node(
    new Dictionary<string, node> {
        {
            "x",
            new node(5)
        },
        {
            "y",
            new node(
                new List<node> {
                    new node(6),
                    new node(new Dictionary<string, node> {}),
                    new node(
                        new Dictionary<string, node> {
                            {
                                "z",
                                new node(new List<node> {})
                            }
                        }
                    ),
                    new node("sedam"),
                }
            )
        },
    }
)
```

Dodatne primjere poziva konstruktora koji se moraju moći kompajlirati možete pogledati na kraju zadaće.

Neka `node` ima dostupnu metodu `pretty_print` koja vraća `string`, koji je jednak, redom po mogućem tipu pohranjenog sadržaja, sljedećim stringovima.

- Uobičajena dekadska reprezentacija broja.
- Pohranjeni string.
- String `"[]"` u slučaju prazne liste, inače započinje znakom `'['` i potom za svaki unutarnji `node` objekt `x` sadrži:
  - prelazak u novi red;
  - praznine koje služe indentaciji (uvlačenju), uvećane za nove 4 praznine;
  - string koji je rezultat poziva `x.pretty_print(...)` gdje na mjestu tri točkice proslijedite potrebne informacije o uvlačenju za slučaj da reprezentacija objekta `x` sadrži više od jedne linije;
  - zarez;

  te završava prelaskom u novi red, prazninama za indentaciju (sad više nema ranije spomenutih dodatnih 4 praznina) i simbolom `']'`.
- String `"{}"` u slučaju praznog rječnika, inače započinje znakom `'{'` i potom za svaki unutarnji par ključa `k` i pripadnog `node` objekta `x` sadrži:

- prelazak u novi red;
- praznine koje služe indentaciji (uvlačenju), uvećane za nove 4 praznine;
- string `k`;
- dvotočku i prazninu;
- string koji je rezultat poziva `x.pretty_print(...)` gdje na mjestu tri točkice proslijedite potrebne informacije o uvlačenju za slučaj da reprezentacija objekta `x` sadrži više od jedne linije;
- zarez;

te završava prelaskom u novi red, prazninama za indentaciju (sad više nema ranije spomenutih dodatnih 4 praznina) i simbolom `'}'`.

Ovako definirana metoda `pretty_print` vrlo je slična uobičajenom ispisu JSON objekata. Glavna razlika je da ovdje postoje suvišni zarezi pred kraj svake neprazne liste i rječnika, te oko ključeva rječnika nema navodnika (to je validan JavaScript kod, ali ne i validan JSON). Poziv na ranije danom primjeru `node` objekta daje sljedeći ispis:

```
{
    x: 5,
    y: [
        6,
        {},
        {
            z: [],
        },
        "sedam",
    ],
}
```

2. dio (5 bodova)

Sada ćemo iskoristiti strukturu `node` za specifikaciju jednostavnih "grafičkih" sučelja. Sučelja će se iscrtavati tekstualno, na način da simbolima "crtamo" grafičke elemente. Sučelja će zapravo biti (stablasta) kolekcija pravokutnih prozora, gdje svaki prozor može sadržavati ili neki tekst, ili nula ili više manjih prozora. Uvijek postoji jedinstveni korijenski prozor, i svaki je prozor rječnik sa sljedećim ključevima.

- `x` (opcionalno): predstavlja x koordinatu (indeks stupca) u kojem započinje ispis ovog prozora, relativno u odnosu na dostupno područje roditeljskog prozora (ako takav postoji). Ako ključ nije u rječniku, pretpostavljamo da je vrijednost 0.
- `y` (opcionalno): predstavlja y koordinatu (indeks retka) u kojem započinje ispis ovog prozora, relativno u odnosu na dostupno područje roditeljskog prozora (ako takav postoji). Ako ključ nije u rječniku, pretpostavljamo da je vrijednost 0.
- `w` (opcionalno): predstavlja širinu prozora, tj. maksimalan broj stupaca kroz koje prolazi ovaj prozor. Ako ključ nije u rječniku, pretpostavljamo da je vrijednost 80.
- `h` (opcionalno): predstavlja visinu prozora, tj. maksimalan broj redaka kroz koje prolazi ovaj prozor. Ako ključ nije u rječniku, pretpostavljamo da je vrijednost 11.
- `z` (opcionalno): predstavlja prioritet prozora prilikom iscrtavanja u slučaju preklapanja prozora. Ako ključ nije u rječniku, pretpostavljamo da je vrijednost 0. Prozore je najjednostavnije iscrtavati krećući od prozora s nižim vrijednostima ovog parametra, pa prema višim vrijednostima. Pritom će neki prozori prebrisati dijelove ranije iscrtanih prozora. Na taj će se način prirodno osigurati ovaj kriterij. Prozore s jednakom `z` vrijednosti iscrtajte u proizvoljnom poretku uz uvjet da prozor koji je sadržan u drugom (roditeljskom) prozoru mora biti iscrtan iznad roditeljskog prozora
- `children` (obavezno): sadržaj prozora. Može biti `node` koji sadrži tekst, ili `node` koji sadrži listu (možda praznu) drugih `node` objekata. U potonjem slučaju svaki `node` objekt u toj listi zapravo je neki manji prozor sadržan u trenutnom prozoru.

Brojevi u rječniku moraju biti zapakirani u `node` objekt, tj. treba koristiti npr.
`new node(new Dictionary<string, node> {{"x", new Node(5)}})` umjesto
`new node(new Dictionary<string, node> {{"x", 5}})`. Za sve navedene cijele brojeve, osim za `z`, možete pretpostaviti da će biti nenegativni cijeli brojevi.

Svaki će prozor u svom prvom i posljednjem retku i stupcu sadržavati simbole koji predstavljaju *rub* prozora: `'-'` (horizontalni rub), `'|'` (vertikalni rub), `'/'` i `'\'` (kutovi). Sadržaj prozora crta se uvijek u području omeđenom rubovima, dakle strogo manjem području od čitavog područja koje prekriva prozor. Prozor koji je sadržan u nekom drugom prozoru (nalazi se u listi pod `children` ključem vanjskog prozora) mora biti čitavom svojom površinom (i rubom) unutar prostora omeđenog rubovima vanjskog prozora. Sve prozore po potrebi treba smanjiti (njihovu visinu i širinu, tj. parametre `w` i `h`) na način da je ispunjen taj uvjet. Taj uvjet vrijedi i za korijenski prozor: njegovi stupci se protežu između (uključivo) 0 i 79, a reci između 0 i 10. Unutarnji prozor kojemu je (nakon eventualne prilagodbe parametara `w` i `h`) visina ili širina manja od 2, ili su mu početne koordinate na rubovima ili potpuno izvan roditeljskog prozora, ne iscrtava se.

U slučaju da prozor sadrži tekst, ispisuje se po dostupnoj širini prozora (bez rubova), počevši od prve dostupne ćelije (koordinate stupca $x+1$ i retka $y+1$). U trenutku ispisa u kojem "kursor"

pređe u redak izvan dostupne površine područja, prekida se ispis. Potrebno je implementirati *word wrap*, tj. riječi koje ne stanu čitave u trenutni redak prebacuju se u sljedeći redak, osim ako je pozicija ispisa (kursora) već bila na početku retka. Vezano uz *word wrap*, treba implementirati i sljedeće ponašanje: sve praznine koji bi bile ispisane na početku reda (osim praznina na početku prvog reda) trebaju biti preskočene.

Za primjer *word wrapa* pretpostavimo da je dan sljedeći prozor dimenzija 5×5 (dostupno područje za sadržane prozore ili tekst je, zbog rubova, samo 3×3):

```
/---\
|   |
|   |
|   |
\---/
```

Primjeri teksta i očekivanog izlaza (u primjerima ulaza znak _ stoji na mjestu praznine):

```
"a_b_c_def"          "__________ab___cd_ef"          "ab_cdefghijk"

/---\                /---\                          /---\
|a b|                |   |                          |ab |
|c  |                |ab |                          |cde|
|def|                |cd |                          |fgh|
\---/                \---/                          \---/
```

Preopteretite metodu `ToString` klase `node` na način da ispišete prozor predstavljen tim objektom kao korijenskim prozorom.

Napomene

- Parametar `z` je *globalan*: npr. ako imamo prozor A sa `z` vrijednosti 3, te A sadrži prozor B sa `z` vrijednosti 2, onda prozor B neće biti vidljiv. Ovo je primjerice različito od `z-index` svojstva u CSS-u koje je lokalno.
- Uočite da stringovi koje neki prozor sadrži ne mogu sadržavati nove linije. To znači da su svi prelasci u nove linije posljedica *word wrapa*.
- Prilikom prilagođavanja širine i visine prozora za prozore za koje se to pokaže potrebnim, nije nužno zapisivati novu širinu i visinu natrag u `node` objekt. Slično, nije potrebno zapisivati stvarne vrijednosti implicitnih parametara. Slično, u redu je ako metoda `pretty_print` ispisuje samo ono što je eksplicitno prisutno u `node` objektu, kako je i opisano u prvom dijelu.

Primjeri

Nekoliko primjera ulaza i izlaza (za oba dijela zadaće), na kojima možete testirati svoje rješenje: primjeri. U oba dijela zadatka bitno je da u potpunosti poštujete specifikaciju. Za provjeru jednakosti vašeg i očekivanog rješenja možete koristiti npr. Diffchecker.com.