Programiranje 2 — tretji izpitni rok
5. september 2022

(1) Napišite program `naloga1.c`, ki ob zagonu z ukazno vrstico

`./naloga1` *vhod*`.pgm` *prag* *izhod*`.pgm`

prepiše slikovno datoteko *vhod*`.pgm` v slikovno datoteko *izhod*`.pgm`, pri čemer vse pike (»piksle«) z vrednostjo *prag* ali več pretvori v pike z vrednostjo 255, vse pike z manjšo vrednostjo pa v pike z vrednostjo 0. Vhodna datoteka je zapisana v sivinskem formatu Netpbm, ki smo ga spoznali na vajah:

`P5`↵
w␣h↵
`255`↵
$p_{0,0}\, p_{0,1} \cdots p_{0,w-1}\, p_{1,0}\, p_{1,1} \cdots p_{1,w-1} \cdots \cdots p_{h-1,0}\, p_{h-1,1} \cdots p_{h-1,w-1}$

V tem formatu sta w in h širina in višina slike (v zapisu ASCII), $p_{i,j}$ pa je 8-bitno število (v dvojiškem zapisu), ki podaja vrednost pike v vrstici z indeksom i in stolpcu z indeksom j. Simbol ␣ predstavlja presledek, simbol ↵ pa znak za prelom vrstice. Izhodna datoteka naj bo zapisana v istem formatu.

Na primer, slika v datoteki `vhod01.pgm` ima širino 3 in višino 2, posamezne pike pa imajo vrednosti 76, 150 in 28 (prva vrstica) ter 226, 104 in 178 (druga vrstica). Če program poženemo z ukazno vrstico

```
./naloga1 vhod01.pgm 150 izhod01.pgm
```

potem mora program ustvariti datoteko `izhod01.pgm`, ki podaja sliko s širino 3 in višino 2 ter pikami z vrednostmi 0, 255 in 0 (prva vrstica) ter 255, 0 in 255 (druga vrstica).

Velja w, $h \in [1, 500]$ in *prag* $\in [0, 256]$. V 50% skritih testnih primerov velja, da ima vsaka pika vhodne slike bodisi vrednost 0 bodisi vrednost 255, obenem pa je *prag* $\in [1, 255]$.

(2) V datoteki `naloga2.h` je podana sledeča deklaracija:

```
typedef struct _Vozlisce {
    char* niz;
    struct _Vozlisce* naslednje;
} Vozlisce;
```

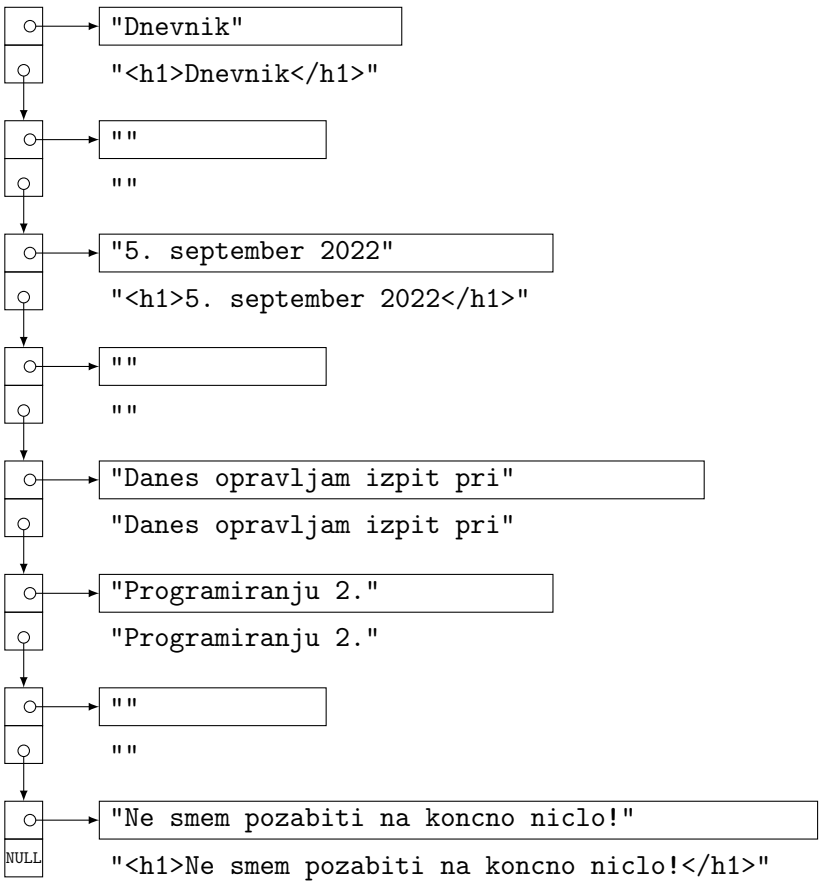
V datoteki `naloga2.c` dopolnite funkcijo

`void vstaviH1(Vozlisce* zacetek)`,

ki sprejme kazalec na prvo vozlišče nepraznega povezanega seznama, v katerem komponente `niz` kažejo na začetke posameznih vrstic besedila. Vrstica je *naslovna*, če ni prazna, obenem pa je tako prejšnja kot naslednja vrstica prazna. (Če katera od teh dveh vrstic ne obstaja, se šteje, kot da je prazna.) Funkcija naj na vsako naslovno vrstico obogati tako, da na njen začetek vstavi niz `<h1>`, na konec pa niz `</h1>`.

V vsaki vrstici je dovolj prostora za oba dodatka (realokacija ni potrebna). Vrstice ne vsebujejo znakov `\n`.

V primeru v datotekah `test01.*` imamo tak seznam (novi nizi so zapisani pod okvirčki z originalnimi nizi):



Vhodni povezani seznam vsebuje kvečjemu 1000 vozlišč, vsaka vrstica pa kvečjemu 1000 znakov. V 50% skritih testnih primerov je vsaka neprazna vrstica tudi naslovna, v 60% *od teh* primerov (torej v 30% od vseh primerov) pa seznam vsebuje natanko eno vozlišče.

(3) V datoteki `naloga3.h` je podana sledeča deklaracija:

```
typedef struct _Vozlisce {
    int podatek;
    struct _Vozlisce* levo;
    struct _Vozlisce* desno;
} Vozlisce;
```

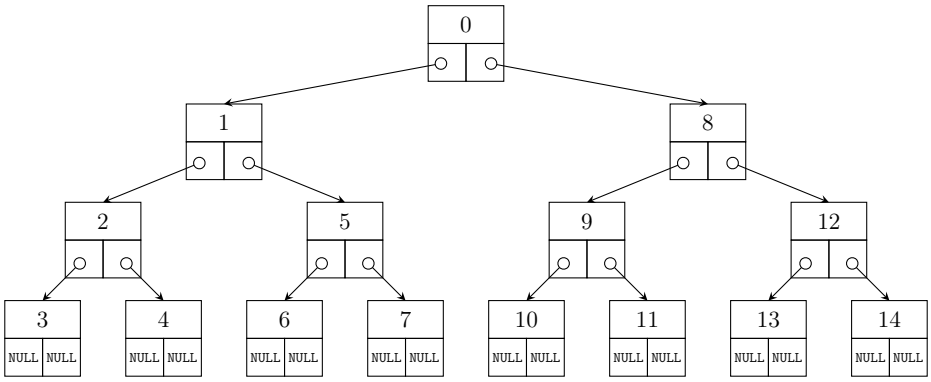
V datoteki `naloga3.c` dopolnite funkcijo

`Vozlisce* drevo(int n, int* podatki)`,

tako da bo zgradila polno dvojiško drevo z `n` nivoji ($1 \leqslant \mathtt{n} \leqslant 20$) in vrnila kazalec na njegov koren. Elemente tabele, na začetek katere kaže kazalec `podatki`, prepišite v komponente `podatek` posameznih vozlišč drevesa, pri čemer

- element z indeksom 0 prepišite v koren;
- podtabelo, ki jo tvorijo elementi z indeksi od 1 do $2^{\mathtt{n}-1} - 1$, rekurzivno prepišite v levo poddrevo korena;
- podtabelo, ki jo tvorijo elementi z indeksi od $2^{\mathtt{n}-1}$ do $2^{\mathtt{n}} - 1$, rekurzivno prepišite v desno poddrevo korena.

Sledeča slika prikazuje drevo za $\mathtt{n} = 4$ in tabelo z vsebino {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14} (`test01.*`):



V 40% skritih testnih primerov so vsi elementi tabele med seboj enaki.