Programiranje 2
Zadaci sa jednostruko povezanim listama sa ranijih ispita

Zadatak 1 *Sadržaj datoteke je aritmetički izraz koji može sadržati zagrade* {, [*i* (. *Napisati program koji učitava sadržaj datoteke* `izraz.txt` *i korišćenjem* **steka** *utvrđuje da li su zagrade u aritmetičkom izrazu dobro uparene. Program štampa odgovarajuću poruku na standardni izlaz.*

Primer 1:
izraz.txt:
```
{[23 + 5344] * (24 - 234)} - 23
```
izlaz:
```
Zagrade su dobro uparene
```

Primer 2:
izraz.txt:
```
{[23 + 5] * (9 * 2)} - {23}
```
izlaz:
```
Zagrade su dobro uparene
```

Primer 3:
izlaz.txt:
```
{[2 + 54) / (24 * 87)} + (234 + 23)
```
izlaz:
```
Zagrade nisu ispravno uparene
```

Primer 4:
izlaz.txt:
```
{(2 - 14) / (23 + 11)}} * (2 + 13)
```
izlaz:
```
Zagrade nisu ispravno uparene.
```

Primer 5:
izlaz.txt:

izlaz:
```
Zagarde jesu ispravno uparene.
```

Zadatak 2 *Lista se učitava sa standardnog ulaza. Elementi liste su celi brojevi i smeštaju se na kraj liste (koristiti datu funkciju* `cvor* ucitaj_listu(char* ime_dat)`*). Nakon učitavanja liste sa standarnog ulaza unose se celi brojevi* `x`, `y` *i* `z`. *Napisati program koji između svakog para brojeva* `x` *i* `y` *umeće broj* `z` *(pri tome* `x` *i* `y` *su uzastopni i* `x` *je pre* `y`*). Ispisati dobijenu listu na standarni izlaz.*
NAPOMENA: Zadatak se mora rešiti korišćenjem listi, u suprotnom broj osvojenih poena je `0`.

Primer 1:
```
4 6 7 2 3 4 6 9 8 4 6 0
4
6
10


4 10 6 7 2 3 4 10 6 9 8 4 10 6
```

Primer 2:
```
 4 5 6 3 4 6 2 6 4 -3 4 89 6 0
 4
 6
 10


 4 5 6 3 4 10 6 2 6 4 -3 4 89 6
```

Primer 3:
```
4 4 2 3 4 4 1 1 4 4 0
4
4
10


4 10 4 2 3 4 10 4 1 1 4 10 4
```

Primer 4:
```
3 3 3 7 0
3
3
10


3 10 3 10 3 7
```

Primer 5:
```
3 3 3 7 0
3
3
3


3 3 3 3 3 7
```

Zadatak 3 *Dve liste se učitavaju sa standardnog ulaza. Elementi liste su celi brojevi i smeštaju se na kraj liste (koristiti datu funkciju* `cvor* ucitaj_listu(char* ime_dat)`*). Izračunati zbir svih elemenata prve liste koji se ne nalaze u drugoj listi.*
NAPOMENA: Zadatak se mora rešiti korišćenjem listi, u suprotnom broj osvojenih poena je `0`.

Primer 1:	Primer 2:	Primer 3:	Primer 4:
3 5 -6 7 8 1 0	0	89 -4 5 78 2 3 21 0	89 5 78 2 21 90 5 6 7 23 7 0
20 100 3 8 9 1 4 0	7 6 5 0	0	76 7 21 100 14 78 0
6	0	194	220

Zadatak 4 *Napisati funkciju* `Cvor* izbaci(Cvor *lista1, Cvor *lista2)` *koja iz* `liste1` *izbacuje sve elemente koji se pojavljuju u* `listi2`*. Testirati funkciju pozivom u main-u, sa standarnog ulaza se učitaju elementi prve liste sve dok se ne unese 0. Potom se učitavaju elementi druge liste sve dok se ne učita 0. Elemente liste dodavati na kraj. Potom pozvati funkciju i novodobijenu listu ispisati na standarni izlaz. Dozvoljeno je pravljenje nove liste.*

Primer 1:	Primer 2:	Primer 3:	Primer 4:
1 3 5 0	3 -90 2 8 7 0	7 8 0	0
3 4 6 7 0	8 3 -4 0	0	10 34 5 67 1 2 0
izlaz: 1 5	izlaz: -90 2 7	izlaz: 7 8	izlaz:

Zadatak 5 *Napisati funkciju* `void umetni(cvor* lista)` *koja izmedju svaka dva elementa u listi umece element koji predstavlja razliku susedna dva.*

Primer 1:	Primer 2:	Primer 3:	Primer 4:
3 4 6 7 0	8 3 -4 0	0	10 34 5 67 1 2 0
3 -1 4 -2 6 -1 7	8 5 3 7 -4		10 -24 34 29 5 -62 67 66 1 -1 2

Zadatak 6 *Napisati funkciju* `void f4(cvor* lista)` *koja iz liste izbacuje svaki drugi element. U listi se nalaze celi brojevi. Lista se unosi sa standardnog ulaza sve dok se ne unese 0.*

Primer 1:	*Primer 2:*	*Primer 3:*	*Primer 4:*
1 2 3 4 5 6 0	*1 2 3 4 5 0*	*34 5 -78 23 0*	*-16 72 13 24 98 0*
1 3 5	*1 3 5*	*34 -78*	*-16 13 98*

Zadatak 7 *Napisati funkciju* `cvor* f2(cvor* L)` *koja dobija glavu liste L kao argument, obrće listu L i vraća novu glavu.*

ulaz: 4->1->2->67->0	*ulaz: 0*	*ulaz: -56->28->31->-1000->0*	*ulaz: 10->9->8->1000000->0*
izlaz: 67 2 1 4	*izlaz:*	*izlaz: -1000 31 28 -56*	*izlaz: 1000000 8 9 10*

Zadatak 8 *Napisati funkciju* `int f7(cvor* lista)` *koja menja elemente liste na sledeći način: ako je tekući element paran, sledeći element uvećava za 1, a ako je element neparan sledeći element smanjuje za 1. Parnost broja se odnosi na tekuću, promenjenu vrednost broja. Funkcija vraća vrednost poslednjeg elementa liste.*

Primer 1:	*Primer 2:*
1->2->3->4->5->NULL	*6->78->2->3->1->NULL*
rez: 4	*rez: 2*

Primer 3:	*Primer 4:*
-87->9->45->2->2->NULL	*22->-34->0->2->1->NULL*
rez: 1	*rez: 0*

Zadatak 9 *Sa standardnog ulaza unosi se lista celih brojeva. Odrediti broj pojavljivanja datog broja u listi.*

Zadatak 10 *Sa standardnog ulaza unosi se lista celih brojeva, a potom se unosi jedan broj. Odrediti poziciju prvog pojavljivanja broja u listi i ispisati na standardni izlaz. Ukoliko broja nema u listi ispisati -1. Pozicija u listi se broji počevši od 1.*

```
Primer 1:                          Primer 2:
ulaz: -20 7 -31 4 5 6 0 -31        ulaz: 4 90 234 -8 0 322

izlaz: 3                           izlaz: -1
-------------------------------------------------------------
Primer 3:                          Primer 4:
ulaz: 9 7 -42 7 343 7 0 7          ulaz: 8 90 8 56 3 2 0 8

izlaz: 2                           izlaz: 1
```

Zadatak 11 *Napisati funkciju koja sažima listu tako što izbacuje svaki element koji se više puta pojavljuje u listi.*

```
Primer 1:
ulaz: 1 3 8 3 1 2 3 6

izlaz: 8 2 6
```

Zadatak 12 *Napisati kôd koji ubacuje novi čvor* `n` *iza čvora liste* `p` *(pretpostavlja se da su* `p` *i* `n` *različiti od* `NULL`*).*

Zadatak 13 *Napisati funkciju* `Cvor* dodaj_u_listu(Cvor *glava, Cvor *novi, int n)` *koja dodaje novi čvor na n-to mesto u listi. Ukoliko lista ima manje od n elemenata novi čvor se dodaje na kraj liste. Kao rezultat funkcija vraća adresu nove glave liste.*

Zadatak 14 *Sa standarnog ulaza se unosi lista celih brojeva. Formirati listu, a potom izbaciti sve one elemente koji su jednaki zbiru svojih suseda i ispisati novodobijenu listu. Ne kreirati novu listu. Smatra se netačnim rešenje u kome se elementi liste samo ispisuju, a lista se pri tome ne menja.*

```
Primer 1:          Primer 2:          Primer 3:          Primer 4:
7 8 1 3 2 1 1 0    23 -4 -27 -23 0    1 2 3 4 5 0        7 7 7 0

7 1 2 1 1          23 -23             1 2 3 4 5          7
```

Zadatak 15 *Napisati funkciju koja iz liste briše sledbenik čvora element, a ukoliko je element NULL, onda briše prvi član liste. Na primer, ako lista L_1 sadrži čvorove 1, 3, 5, 7, i ukoliko element ukazuje na čvor 3 briše se njegov sledbenik i lista sadrži čvorove 1, 3, 7. Ako lista L_1 sadrži čvorove 1, 3, 4, 5, i ukoliko je element NULL, briše se prvi član liste i lista sadrži čvorove 3, 4, 5.*

Zadatak 16 *Lista se učitava sa standardnog ulaza. Elementi liste su celi brojevi i učitavaju se sve dok se ne unese* `0` *i smeštaju se na kraj liste (koristiti datu funkciju* `cvor* ucitaj_listu()`*). Iz date liste izbaciti sve elemente čija vrednost je neparan broj. Dobijenu listu ispisati na standardni izlaz. NAPOMENA: Zadatak se mora rešiti korišćenjem listi, u suprotnom broj osvojenih poena je* `0`*.*

Primer 1:	**Primer 2:**	**Primer 3:**	**Primer 4:**	**Primer 5:**
2 3 6 7 8 91 0	5 7 9 20 24 56 78 0	24 3 7 93 12 4 11 0	9 7 5 3 1 0	6 8 10 0
2 6 8	20 24 56 78	24 12 4		6 8 10

Zadatak 17 *Lista se učitava sa standardnog ulaza. Elementi liste su celi brojevi i učitavaju se sve dok se ne unese* `0` *i smeštaju se na kraj liste (koristiti datu funkciju* `cvor* ucitaj_listu()`*). Nakon unosa elemenata liste, unosi se ceo broj* `k`*. Iz date liste izbaciti sve elemente koji su deljivi sa* `k`*. Dobijenu listu ispisati na standardni izlaz. U slučaju greške (pokušaj deljenja sa* `0`*) ispisati -1. NAPOMENA: Zadatak se mora rešiti korišćenjem listi, u suprotnom broj osvojenih poena je* `0`*.*

Primer 1:	**Primer 2:**	**Primer 3:**	**Primer 4:**
2 3 6 7 8 90 0 3	5 7 9 20 24 56 78 0 5	24 3 7 93 12 4 11 0 0	9 7 5 3 1 0 100
2 7 8	7 9 24 56 78	-1	9 7 5 3 1