Programiranje 2
Jednostruko povezane liste

Zadatak 1 *Napisati biblioteku za rad sa jednostruko povezanim listama koje sadrže cele brojeve.*

1. *Definisati strukturu* `Cvor` *koja sadrži ceo broj* `vrednost` *i pokazivač na* `sledeći` *čvor liste.*
2. *Napisati funkciju* `Cvor* napravi_cvor(int broj);` *koja kao povratnu vrednost vraća čvor čija je vrednost prosledjeni broj.*
3. *Napisati funkciju* `void dodaj_na_pocetak(Cvor** lista, int* broj);` *koja pravi novi čvor čija je vrednost prosledjeni broj i postavlja taj čvor na početak prosledjene liste.*
4. *Napisati funkciju* `Cvor* dodaj_na_pocetak2(Cvor* glava, int br)` *koja dodaje element na početak liste i vraća glavu nove liste.*
5. *Napisati funkciju* `void ispisi_listu(Cvor* lista, FILE* f);` *koja ispisuje prosledjenu listu u datoteku* `f`*. Pogledati format ispisa u primeru ispod. Funkciju implementirati i rekurzivno i iterativno.*
6. *Testirati prve 4 stavke, tako što se sa standardno ulaza učitava broj n, a zatim n brojeva. Brojeve dodavati redom na pocetak liste. Listu ispisati na standardni izlaz.*

```
Primer1:                Primer 2:                   Primer 3:
5                       7                           -5
1 2 3 4 5               12 10 56 -98 15 2 100       -1

[5, 4, 3, 2, 1]         [100, 2, 15, -98, 56, 10, 12]
```

7. *Napisati funkciju* `void dodaj_na_kraj(Cvor** lista, int broj);` *koja pravi novi čvor čija je vrednost prosledjeni broj i dodaje taj čvor na kraj prosledjene liste.*
8. *Napisati funkciju* `Cvor* dodaj_na_kraj2(Cvor* lista, int br)` *koja dodaje element na kraj liste. Funkciju implementirati i rekurzivno i iterativno.*
9. *Napisati funkciju* `void ucitaj_listu(Cvor** lista, FILE* f);` *koja iz datoteke* `f` *učitava brojeve sve dok ne dodje do kraja datoteke i za svaki učitani broj pravi novi čvor i dodaje ga na kraj liste.*
10. *Napisati funkciju* `void oslobodi_listu(Cvor* lista);` *koja oslobađa memoriju koju zauzima prosledjena lista. Funkciju implementirati i rekurzivno i iterativno.*
11. *Napisati program koji testira rad ovih funkcija. Ime datoteke se zadaje kao argument komandne linije. Ime datoteke može biti i* `stdin` *i u tom slučaju kraj unosa se završava sa* `CTRL+D`*.*

```
Primer1:                Primer 2:
1 2 3 4 5               12 10 56 -98 15 2 100

[1, 2, 3, 4, 5]         [12, 10, 56, -98, 15, 2, 100]
```

Ova biblioka će biti na raspolaganju studentima u svakom ispitnom roku.

Zadatak 2 *Napisati funckiju* `int veci_od(Cvor* lista, int broj);` *koja proverava koliko ima čvorova u listi čija je vrednost veća od prosledjenog broja. Iz datoteke koja se zadaje kao argument komandne linije se učitavaju elementi liste. Sa standardnog ulaza se učitava broj x. Na standardni izlaz ispisati rezultat izvršavanja napisane funkcije. U slučaju greške na standardni izlaz za greške ispisati -1.*
Napomena: Koristiti biblioteku za rad sa listama.

```
Primer 1:
./a.out ulaz.txt

ulaz1.txt
1 4 8 90 999 4 -34 1 3 7 391 234

300

2
```

Zadatak 3 *Napisati funkciju* `_cvor* dodaj_element(_cvor* lista, int br)` *koja dodaje element u neopadajuće sortiranu listu tako da lista ostane sortirana. Testirati funkciju pozivom u* `main-u`.

```
Primer 1:
./a.out lista.txt
lista.txt:
-5 1 5 6 8 9 18
Unesi element:
12

Nova lista:
[-5, 1, 5, 6, 8, 9, 12, 18]
```

```
Primer 2:
./a.out stdin
200 300 400 500
Unesi element:
-100

Nova lista:
[-100, 200, 300, 400, 500]
```

Zadatak 4 *Napisati funkcije za izbacivanje elementa iz liste:*

a) `_cvor* izbaci1(_cvor* lista, int br)` *koja izbacuje prvo pojavljinjanje datog elementa u listi i vraća glavu nove liste. (prokomentarisati zašto bi moralo da se piše* `void izbaci1(_cvor** lista, int br)` *u slučaju kada bi pravili funkciju koja nema povratnu vrednost).*

```
Primer 1:
./a.out dat1.txt
dat1.txt:
1 56 7 23 7 9 -90
Unesi element:
7

izlaz:
[1, 56, 23, 7, 9, -90]
```

```
Primer 2:
./a.out dat1.txt
dat1.txt:
880 -4 57 880 3 45 880 45
Unesi element:
880

izlaz:
[-4, 57, 880, 3, 45, 880, 45]
```

b) `_cvor* izbaci2(_cvor* lista, int br)` *koja izbacuje poslednje pojavljinjanje datog elementa u listi i vraća glavu nove liste.*

```
Primer 1:
./a.out dat1.txt
dat1.txt:
1 56 7 23 7 9 -90
Unesi element:
7

izlaz:
[1, 56, 7, 23, 9, -90]
```

```
Primer 2:
./a.out dat1.txt
dat1.txt:
880 -4 57 880 3 45 880
Unesi element:
880

izlaz:
[880, -4, 57, 880, 3, 45]
```

c) `_cvor* izbaci3(_cvor* lista, int br)` *koja izbacuje svako pojavljinjanje datog elementa u listi i vraća glavu nove liste.*

```
Primer 1:
./a.out dat1.txt
dat1.txt:
1 56 7 23 7 9 -90
Unesi element:
7

izlaz:
[1, 56, 23, 9, -90]
```

```
Primer 2:
./a.out dat1.txt
dat1.txt:
880 -4 57 880 3 45 880 45
Unesi element:
880

izlaz:
[-4, 57, 3, 45, 45]
```

Zadatak 5 *Date su dve jednostruko povezane liste L_1 i L_2. Napisati funkciju koja od ovih listi formira novu listu L koja sadrži naizmenično raspoređene čvorove listi L_1 i L_2: prvi čvor iz L_1, prvi čvor iz L_2, drugi čvor L_1, drugi čvor L_2, itd. Ne formirati nove čvorove, već samo postojeće rasporediti u jednu listu. Prva lista se učitava iz datoteke čije se ime zadaje kao prvi argument komandne linije, a druga iz datoteke čije se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz. Ne praviti nove čvorove liste, već ispremeštati postojeće.*

```
Primer 1:
./a.out dat1.txt dat2.txt
dat1.txt:
98 -200 203 400
dat2.txt:
78 300 -301 402

izlaz:
[98, 78, -200, 300, 203, -301, 400, 402]
```

```
Primer 2:
./a.out jedan.txt dva.txt
jedan.txt:
65 21 18 78 102 203 94
dva.txt:
100 23 78

izlaz:
[65, 100, 21, 23, 18, 78, 78, 102, 203, 94]
```

Zadatak 6 *Napisati funkciju* `_cvor* spoji(_cvor* lista1, _cvor* lista2)` *koja spaja dve neopadajuće sortorane liste u treću, takođe neopadajuće sortiranu listu. Funkcija ne sme da kreira nove čvorove (već samo da preraspodeli postojeće). Testirati funkciju pozivom u* `main-u`*. Prva lista se nalazi u datoteci čije ime je prvi argument komandne linije, a druga lista se nalazi u datoteci čije ime je drugi argument komandne linije.*

```
Primer 1:
./a.out jedan.txt dva.txt
jednan.txt:
98 200 203 400
dva.txt:
78 300 301 402

izlaz:
[78, 98, 200, 203, 300, 301, 400, 402]
```

```
Primer 2:
./a.out jedan.txt dva.txt
jedan.txt:
18 21 65 78 94 102 203
dva.txt:
23 78 100

izlaz:
[18, 21, 23, 65, 78, 78, 94, 100, 102, 203]
```

Zadatak 7 *Napisati funkciju koja u datoj listi izmedju svaka dva elementa čiji su zbir ili razlika jednaki datom broju k umeće -1. Glavni program učitava listu sa* `stdin` *i ceo broj k. Potrebno je ispisati rezultujuću listu na stadardni izlaz. Nije dozvoljeno korišćenje pomoćne liste. Ne analizirati prvi i poslednji element liste jer oni nemaju oba suseda.*
U slučaju greške na standardni izlaz za greške ispisati -1.
Napomena: Koristiti biblioteku za rad sa listama.

```
Primer 1:
1 2 3 1 2 3
3

[1, -1, 2, 3, 1, -1, 2, 3]
```

```
Primer 2:
4 2 1 5 6 2 4
2

[4, -1, 2, 1, 5, 6, 2, -1, 4]
```

```
Primer 3:
1 3 1 1 3 1
2

[1, -1, 3, -1, 1, -1, 1, -1, 3, -1, 1]
```

```
Primer 4:

5
```

Zadatak 8 *Napisati funkciju koja u datoj listi izbacuje susedne elemente čiji je zbir jednak datom broju k. Potrebno je ispisati tako dobijenu listu na stadardni izlaz. Nije dozvoljeno korišćenje pomoćne liste. Nije dovoljno samo ispisati traženu listu već je potrebno elemente zaista izbaciti i konstruisati novu listu. Elementi liste su celi brojevi, lista se unosi sa standarnog ulaza. Nakon unosa elemenata liste unosi se broj k. U slučaju greške na standardni izlaz za greške ispisati -1.*
Napomena: Koristiti biblioteku za rad sa listama.

Primer 1:	**Primer 2:**	**Primer 3:**	**Primer 4:**
13 4 5 10 0	13 4 5 9 9 0	13 4 5 4 3 0	4 5 3 -2 11 -2 11 -2 0
9	9	9	9
[13, 10]	[13, 9, 9]	[13, 3]	[3]