Programiranje 2
Dinamička alokacija memorije

1 ZADACI SA ČASA

Zadatak 1 *Napisati program koji sa standardnog ulaza učitava dimenziju niza celih brojeva, a zatim i njegove elemente. Ne praviti pretpostavke o dimenziji niza. Na standardni izlaz ispisati učitane brojeve u obrnutom poretku. U slučaju greške na standardni izlaz za greške ispisati* `-1`.

```
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
```

Zadatak 2 *Napisati program koji sa standardnog ulaza učitava niz razlomaka. Prvo se učita broj razlomaka, a zatim i razlomci u obliku brojilac imenilac. Na standardni izlaz ispisati sve razlomke čija je vrednost veća od prosečne vrednosti svih učitanih razlomaka, u obliku brojilac/imenilac. Ne praviti nikakve pretpostavke o maksimalnoj dimenziji niza. U slučaju greške na standardni izlaz za greške ispisati* `-1`.

```
Primer1:                Primer 2:
5                       3
1 2                     -5 6
3 4                     100 31
5 7                     2 9
2 3
6 5

6/5                     100/31
```

Zadatak 3 *Napisati program koji učitava niz celih brojeva iz datoteke brojevi.txt sve do unosa broja 0, koristeći funkciju za realokaciju memorije sa korakom k (koji se zadaje kao argument komandne linije). Na standardni izlaz ispisati sve brojeve koji su veći od središnjeg elementa niza. U slučaju parne dimenzije niza, središnji element računati kao aritmetičku sredinu dva elementa najbliža sredini. U slučaju greške na standardni izlaz za greške ispisati* `-1`.

```
Primer1:                Primer 2:
./a.out 10              ./a.out 5
brojevi.txt             brojevi.txt
1 2 3 4 5 0             12 10 56 -98 15 2 100 55 0

4 5                     12 10 56 15 2 100 55
```

Zadatak 4 *Napisati program koji sa standardnog ulaza učitava matricu celih brojeva. Prvo se učitaju broj vrsta i kolona matrice, a zatim i elementi matrice. Na standardni izlaz ispisati učitanu matricu. Zatim napisati funkciju*

`int sum_max(int **A, int n, int m)`

koja računa zbir najvećih elemenata u svakoj vrsti. Ispisati rezultat izvršavanja funkcije na standardni izlaz. U slučaju greške na standardni izlaz za greške ispisati `-1`.

Primer1:
```
2 3
1 2 3
4 5 6


1 2 3
4 5 6
9
```

Primer 2:
```
3 3
-5 6 6
100 31 10
2 9 -55

-5 6 6
100 31 10
2 9 -55
115
```

Zadatak 5 *Sa standardnog ulaza se učitava reč pretrage, dimenzija niza, a zatim i niz reči. Pretpostavljati da je maksimalna duzina reči 20 karaktera (prostor za reč alocirati dinamički). Na standardni izlaz ispisati indeks prvog i poslednjeg pojavljivanja tražene reči u okviru unetog niza reči. U slučaju da se reč ne pojavljuje u nizu, ispisati -1 za obe pozicije. U slučaju greške na standardni izlaz za greške ispisati* `-1`.

Primer 1:
```
kisa
9
danas je padala kisa a sutra kisa nece padati

3 6
```

Primer 2:
```
programiranje
3
mi volimo matematiku

-1 -1
```

Zadatak 6 *Definisati strukturu*

```
typedef struct{
    unsigned int a, b;
    char ime[5];
} PRAVOUGAONIK;
```

kojom se opisuje pravougaonik dužinama svojih stranica i imenom. Napisati program koji iz datoteke čije ime se zadaje kao argument komandne linije učitava pravougaonike (nepoznato koliko), a zatim prvo ispisuje imena onih pravougaonika koji su kvadrati, a nakon toga ispisuje vrednost najveće površine medju pravougaonicima koji nisu kvadrati. U slučaju greške na standardni izlaz za greške ispisati `-1`.

Primer 1:
```
./a.out pravougaonici.dat
pravougaonici.dat:
2 4 p1
3 3 p2
1 6 p3

p2 8
```

Primer 2:
```
./a.out dva.dat
dva.dat:
5 2 pm
4 7 pv

28
```

Primer 3:
```
./a.out tri.dat
tri.dat:
5 5 m
3 3 s
8 8 xl

m s xl
```

Primer 4:
```
./a.out primerx.dat
primerx.dat:
9 7 p

63
```

Primer 5:
```
./a.out prazna.dat
prazna.dat:
```

Zadatak 7 *Definisati strukturu* `STUDENT` *koja sadrži:*

- `puno_ime` *(u polju se čuva ime i prezime studenta, npr. "Marko Markovic", maksimalna dužina polja je 100 karaktera),*
- `ocene` *(sadrži najviše 10 ocena studenta)*
- `broj_ocena` *(ukupan broj ocena za studenata)*

- `prosek` *(prosečna ocena)*

U datoteci čiji se naziv zadaje kao argument komandne linije se nalaze podaci o studentima (prvo broj studenata, a zatim i i informacije o svakom studentu). Za svakog studenta unosi se ime i prezime razdvojeno razmakom, a potom ocene koje se završavaju sa 0. Napisati funkciju

```
int najveci_prosek(STUDENT* niz, int n)
```

koja pronalazi studenta sa najvećim prosekom i vraća poziciju u nizu na kojoj se taj student nalazi. Napisati funkciju

```
void ispisi(const STUDENT* s)
```

koja ispisuje podatke o studentu u formatu: ime prezime, prosek na dve decimale. Testirati obe funkcije pozivom u glavnom programu, tako što prvo treba naći studenta sa najvećim prosekom, a zatim ispisati informacije o tom studentu. U slučaju greške na standardni izlaz za greške ispisati `-1`.

Primer 1:
```
./a.out studenti.txt
studenti.txt:
4
Marko Markovic 5 6 7 8 9 0
Jelena Jankovic 10 10 10 0
Filip Viskovic 10 9 8 7 6 0
Jana Peric 10 10 9 9 8 8 7 7 0

Jelena Jankovic 10.00
```

Primer 2:
```
./a.out

-1
```

Primer 3:
```
./a.out prazna.dat
prazna.dat:
```

Zadatak 8 *Napisati program koji sa standardnog ulaza učitava dimenziju niza celih brojeva, a zatim i njegove elemente. Na standardni izlaz za svaki broj ispisati koliko ima brojeva koji se nalaze ispred njega u nizu, a koji su manji od tog broja. U slučaju greške na standardni izlaz za greške ispisati* `-1`.

Primer1:
```
5
1 2 3 4 5

0 1 2 3 4
```

Primer 2:
```
7
12 10 56 -98 15 2 100

0 0 2 0 3 1 6
```

Primer 3:
```
-5
-1
```

Zadatak 9 *Ime datoteke dato je kao argument komandne linije. U datoteci se nalaze otvorene i zatvorene zagrade i još nekakav tekst. Napisati program koji proverava da li su zagrade pravilno uparene. Npr.* `ab( cd) ..` *odgovor je* `jesu`*, a* `..)ba()` *odgovor je* `nisu`*. U slučaju greške na standardni izlaz za greške ispisati* `-1`.

Primer 1:
```
./a.out zagrade.txt
zagrade.txt:
ab( cd) ..
((3+4)*5+1)*9

jesu
```

Primer 2:
```
./a.out primer2.dat
primer2.dat:
(7+8
nisu(
uparene

nisu
```

Primer 3:
```
./a.out primer3.dat
primer3.dat:
)) 7 + 6 ((

nisu
```

Primer 4:
```
./a.out

-1
```

Zadatak 10 *Skupovi karaktera*

a) Napisati C funkciju `int unesiSkup(char **s, FILE* f)` *kojom se unosi skup elemenata iz datoteke F. Skup se predstavlja kao niz karaktera, pri čemu su dozvoljeni elementi skupa mala i velika slova abecede, kao i cifre. Unos se prekida kada se naiđe na znak za novi red ili nedozvoljeni karakter za skup Maksimalan broj elemenata skupa nije poznat. Funkcija vraća broj elemenata skupa koji su uspesno učitani.*

b) Napisati funkciju `void prebroj(char *s, int *br_slova,int *br_cifara)` *kojom se određuje broj slovnih elemenata skupa (velikih ili malih slova) kao i broj cifara u skupu.*

c) Napisati glavni program gde se unose podaci o skupu elemenata. Ime datoteke se zadaje kao argument komandne linije. Na stadardni izlaz ispisati informacije o broju slova i cifara (koristiti funkcije pod a) i b)).

U slučaju greške na standardni izlaz za greške ispisati `-1`.

Primer 1:
```
./a.out skup.txt
skup.txt:
abc56ighj9012hjFGHH

broj slova: 13
broj cifara: 6
```

Primer 2:
```
./a.out

-1
```

Primer 3:
```
./a.out skup2.txt
skup2.txt:
ovdeimamo$dolar

broj slova: 9
broj cifara: 0
```

Primer 4:
```
./a.out skup3.txt
skup3.txt:
broJ3
broj5

broj slova: 4
broj cifara: 1
```

Zadatak 11 *Definisati strukturu*

```
typedef struct{
    int x;
    int y;
    int z;
} VEKTOR;
```

kojom se opisuje trodimenzioni vektor. U datoteci `vektori.txt` *nalazi se nepoznati broj vektora. Na standarni izlaz ispisati koordinate vektora sa najvećom dužinom. Dužina vektora se izračunava po formuli:*

$$|v| = \sqrt{x^2 + y^2 + z^2}$$

U slučaju greške na standardni izlaz za greške ispisati `-1`.

Primer 1:	***Primer 2:***	***Primer 3:***	***Primer 4:***
2 *4 -1 7* *3 1 2* *4 -1 7*	*-9* *-1*	*3* *0 0 0* *0 1 0* *1 0 0* *0 1 0*	*4* *3 0 1* *4 5 2* *1 0 0* *2 -1 2* *4 5 2*