Zadatak 1. *Napisati program koji ispisuje broj navedenih argumenata komadne linije, a zatim i same argumente sa rednim brojevima.*

Zadatak 2. *Ako su celi brojevi* `a` *i* `b` *argumenti komandne linije napraviti niz* `A[0] = a, A[1] = a+1, A[2] = a+2, ..., A[b-a] = b` *i ispisati ga.*

Zadatak 3. *Uobičajena praksa na UNIX sistemima je da se argumenti komandne linije dele na opcije i argumente u užem smislu. Opcije počinju znakom - nakon čega obično sledi jedan ili više karaktera koji označavaju koja je opcija u pitanju. Ovim se najčešće upravlja funkcionisanjem programa i neke mogućnosti se uključuju ili isključuju. Argumenti načšće predstavljaju opisne informacije poput na primer imena datoteka. Napisati program koji ispisuje sve opcije koje su navedene u komandnoj liniji. Na primer, ako se program pokrene sa*
`./echoopts -abc input.txt -d -Fg output`
program treba da ispiše

```
Prisutne su opcije:
a b c d F g
```

Zadatak 4. *Kao argumenti komandne linje zadate su dimenzije matrice* `A` *(*`m` *i* `n`*). Element matrice se naziva sedlo ako je istovremeno najmanji u svojoj vrsti, a najveći u svojoj koloni. Ispisati indekse onih elemenata matrice koji su sedlo.*

Zadatak 5. *Napisati program koji poredi dva fajla i ispisuje redni broj linija u kojima se fajlovi razlikuju. Imena fajlova se zadaju kao argumenti komandne linije.*

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 ispisuje imena onih pravougaonika koji su kvadrati i vrednost najveće površine medju pravougaonicima koji nisu kvadrati. U slučaju unosa nekorektnih dužina stranica pravougaonika ili nekorektne vrednosti broja `n`*, ispisati -1 i odmah prekinuti izvršavanje programa. Maksimalan broj pravougaonika je 200.*

```
Primer 1:   Primer 2:    Primer 3:   Primer 4:   Primer 5:
3           2            3           1           0
2 4 p1      5 2 pm       5 5 m       9 7 p
3 3 p2      4 7 pv       3 3 s
1 6 p3                   8 8 xl

p2 8        28           m s xl       63         -1
```

Zadatak 7. *Ime datoteke dato je kao argument komandne linije. U datoteci se nalaze otvorene i zatvorene zagrade i još nekakav tekst. Proveriti da li su zagrade pravilno uparene. Npr.* `ab(  cd) ..` *odgovor je* `jesu`*, a* `..)ba()` *odgovor je* `nisu`*.*

Zadatak 8. *Napraviti strukturu* `STUDENT` *koja sadrži:*

- `ime` *(u polju se čuva ime i prezime studenta, napr. "Marko Markovic", maksimalna dužina polja je 100 karaktera),*
- `oc` *(sadrži najviše 10 ocena studenta)*
- `br_ocena` *(ukupan broj ocena za studenata)*
- `pr_oc` *(prosečna ocena)*

U datoteci se nalaze podaci o studentima. Za svakog studenta unosi se ime i prezime razdvojeno razmakom (uputstvo: može se korisiti `gets`*), a potom u sledećem redu ocene koje se završavaju sa 0. Pronaći studenta koji ima najveći prosek i ispisati sve njegove podatke. Maksimalan broj studenta je 100.*

Zadatak 9. *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 je 1000). 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)).

Zadatak 10. *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 (maksimalno ih može biti 200). Ucitati ih u niz i ispisuje na standardnom izlazu koordinate vektora sa najvećom dužinom. Dužina vektora se izračunava po formuli:*

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

U slučaju greške ispisati -1 i prekinuti izvršavanje programa.

Primer 1:	Primer 2:	Primer 3:	Primer 4:
2	67	3	4
4 -1 7		0 0 0	3 0 1
3 1 2		0 1 0	4 5 2
		1 0 0	1 0 0
			2 -1 2
4 -1 7	-1	0 1 0	4 5 2