U svim zadacima vrijedi sljedeće:

- glavni program mora biti smješten u modulu **glavni.c**
- funkcije moraju biti smještene u modulu **funkcije.c**
- prototipovi (deklaracije) funkcija moraju biti smješteni u datoteci **funkcije.h**

Nije dopušteno korištenje statičkih, eksternih varijabli te naredbe goto osim tamo gdje je njihovo korištenje nužno.

1. zadatak

a) Napisati funkciju `stupnjeviURadijane` koja kao argument prima decimalni broj - kut izražen u stupnjevima, a kao rezultat vraća kut izražen u radijanima.

b) Napisati funkciju `izracunajUdaljenost` koja kao argumente prima geografsku širinu: *lat1* i *lat2* i dužinu: *long1* i *long2* dvije točke na površini Zemlje, a kao rezultat vraća njihovu udaljenost `d` u kilometrima izračunatu pomoću Haversinove formule:

$$a = sin^2\left((lat2 - lat1)/2\right) + \cos(lat1) * \cos(lat2) * sin^2\left((long2 - long1)/2\right)$$

$$d = R * 2 * atan2(\sqrt{a}, \sqrt{1-a}) \quad (R = 6{,}371 \text{ prosječna vrijednost radijusa Zemlje})$$

Argumenti trigonometrijskih funkcija u gornjim izrazima moraju biti izraženi u radijanima. Za računanje vrijednosti trigonometrijskih funkcija sinus, kosinus i arkus-tangens koristiti funkcije `sin`, `cos` i `atan2` iz biblioteke `math.h`.

c) U glavnom programu je potrebno s tipkovnice učitati ukupan broj gradova (ne veći od 100 niti manji od 3). Ukupan broj gradova potrebno je učitavati sve dok se ne učita ispravna vrijednost. Potom je potrebno učitati podatke za zadani broj gradova: poštanski broj grada i njegove GPS koordinate. Za koordinate grada se učitavaju 2 realna broja (kutovi izraženi u stupnjevima): *latitude* ∈ [-180, 180], specificira sjever-jug i *longitude* ∈ [-90, 90], specificira istok-zapad poziciju na površini Zemlje. Potom je potrebno odrediti i ispisati udaljenosti između svaka 2 grada u kilometrima. Obavezno koristiti funkcije iz a) i b) dijela zadatka.

Primjer ulaznih podataka i izlaznog rezultata:

```
Upisite broj gradova (od 3 do 100): 2
Upisite broj gradova (od 3 do 100): 3

Upisite postanski broj i koordinate 1. grada: 10000 45.81497 15.97851
Upisite postanski broj i koordinate 2. grada: 21000 43.50692 16.44245
Upisite postanski broj i koordinate 3. grada: 23000 44.114934 15.228952

Medjusobne udaljenosti gradova:
10000 - 21000: 259.251953 km
10000 - 23000: 198.017715 km
21000 - 23000: 118.540222 km
```

2. zadatak

Linearni kongruentni generator je generator niza pseudoslučajnih cijelih brojeva. Generator pri svakom pozivu vraća sljedeći član niza pseudoslučajnih brojeva X_{i+1}, kojeg izračunava na temelju prethodnog člana niza X_i.

$$X_{i+1} = (A \cdot X_i + C) \bmod M$$

Cjelobrojne konstante `A`, `C` i `M` se unaprijed zadaju, a član niza X_0, tj. početni član niza pseudoslučajnih brojeva koji se naziva sjeme ili *seed*, generatoru se zadaje neposredno prije početka generiranja niza pseudoslučajnih brojeva.

a) Napisati funkciju `setSeed` kojom se generatoru zadaje početni član niza X_0 (tj. sjeme ili *seed* generatora). Funkcija kao ulazni argument prima nenegativni cijeli broj, a ne vraća rezultat. Funkciju napisati tako da, ako se ova funkcija ne pozove prije početka generiranja niza, tada X_0 treba imati pretpostavljenu (ili *default*) vrijednost 1.

b) Napisati funkciju `getRand` koja nema ulaznih argumenata, a kao rezultat daje nenegativni cijeli broj koji se izračunava na temelju prethodnog člana niza (prema gore navedenom izrazu). Za konstante u izrazu koristiti sljedeće vrijednosti:

`A = 9001; C = 29; M = 225;` (uočite da će za ove parametre biti generirani isključivo brojevi iz intervala [0, 224])

Vrijednost za početni član X_0 ne smije biti dostupna (ne smije biti u dosegu) iz glavnog programa, tj. ne može se niti čitati niti postavljati niti na jedan drugi način nego funkcijom `setSeed`.

Primjer: Izvršavanjem sljedećeg odsječka programa

```
setSeed(1000);
for (i = 1; i <= 5; i++)
   printf("%u  ", getRand());
```

dobio bi se rezultat: `129  158  187  216  20`

Izvršavanjem istog odsječka, ali bez pozivanja funkcije `setSeed`, dobio bi se rezultat: `30  59  88  117  146`

c) Napisati glavni program koji će s pomoću generatora pseudoslučajnih brojeva generirati rečenicu. S tipkovnice učitati sjeme za generator i pozivom funkcije setSeed inicijalizirati generator. Generirati slučajni broj koji će predstavljati duljinu rečenice. Zatim generirati znakove od kojih će rečenica biti sastavljena tako da se generatorom generiraju slučajne ASCII vrijednosti. Ako je generirana ASCII vrijednost znaka koji je malo ili veliko slovo ili zarez ili praznina, iskoristiti tu vrijednost za sljedeći znak rečenice. Generirane ASCII vrijednosti koje ne zadovoljavaju taj uvjet treba preskočiti. Rečenicu završiti točkom. Sjeme za generator učitavati, inicijalizirati generator, generirati te na zaslon ispisivati rečenice, sve dok se za sjeme generatora ne upiše broj nula.

Primjer ulaznih podataka i izlaznog rezultata:

```
Upisite sjeme: 7000
Sp ZwDaKhRoYvCJgQnXuB,IfPmWtAHeOlVszGdNkUryFcMjTqxEbLi.

Upisite sjeme: 200
xEbL.

Upisite sjeme: 1
XuB,IfPmWtAHeOlVszGdNkUryFcMjT.

Upisite sjeme: 0
```

3. zadatak

a) Napisati funkciju koja računa uniju dva ulazna skupa prirodnih brojeva i funkciju koja provjerava je li jedan skup prirodnih brojeva podskup drugog. Prototipovi funkcija su:

```
void unija(int n1, int *s1, int n2, int *s2, int nrez, int *rez);
int je_podskup(int npodskup, int *podskup, int nskup, int *skup);
```

b) Napisati glavni program u kojem se s tipkovnice učitavaju brojevi u ulazne skupove prirodnih brojeva (prvo u `skup1` pa u `skup2`, a učitavanje prestaje kada se unese broj koji nije prirodan, npr. -1). Paziti na svojstvo skupova da ne sadrže duplikate - ako se učita duplikat, ne uključiti ga u skup! Pretpostaviti da ulazni skupovi neće imati više od 20 elemenata. Program, nakon učitavanja elemenata skupova, pomoću funkcija iz a) dijela zadatka prvo izračunava uniju, zatim provjerava je li prvi skup podskup drugog i obrnuto te ispisuje rezultat svake od operacija.

Npr. za ulazne podatke:

```
Skup 1: 1,2,4,5,6,7,8
Skup 2: 1,2,5
```

Program treba ispisati:

```
Unija: 1 2 4 5 6 7 8
Skup 1 NIJE podskup skupa 2
Skup 2 JE podskup skupa 1
```

4. zadatak

a) Napisati funkciju čiji je prototip:

```
void generirajNiz(char *znakovi, int duljinaNiza, char *genNiz);
```

Funkcija treba od znakova koji se nalaze u nizu `znakovi`, formirati niz `genNiz` zadane duljine `duljinaNiza`, na sljedeći način:

- ako generirani niz mora biti dulji od zadanog niza, redom nadodavati znakove iz zadanog niza potreban broj puta. Npr. za zadani niz `Ac12` i traženu duljinu niza 7, funkcija generira niz `Ac12Ac1`.
- ako generirani niz mora biti jednake duljine ili kraći od zadanog niza, tada generirani niz sadrži zadnjih `duljinaNiza` znakova zadanog niza. Npr. za zadani niz `Ac12` i traženu duljinu niza 3, funkcija generira niz `c12`.

b) Napisati funkciju `dobarNiz` koja za zadani niz znakova vraća vrijednost 1 (cijeli broj), ako se u njemu isti znak ne pojavljuje više puta, a u suprotnom vraća vrijednost 0. Npr. za zadani niz znakova `a1a3d`, funkcija vraća 0, a za niz `a13d` vraća 1.

c) U glavnom programu potrebno je pomoću tipkovnice učitati niz znakova (koji sigurno nije dulji od 20 znakova - nije potrebno provjeravati) te prirodni broj, koji predstavlja duljinu niza koji je potrebno generirati i ne smije biti veći od 100. Niz znakova je potrebno učitavati sve dok se ne učita niz u kojem se znakovi ne ponavljaju, a duljinu niza sve dok se ne unese ispravna vrijednost. Nakon toga funkcijom `generirajNiz` generirati niz i na zaslon ispisati učitani i generirani niz.

5. zadatak

a) Napisati funkciju `prviZnak` koja pronalazi prvo pojavljivanje nekog od znakova iz zadanog niza znakova `niz1` u zadanom nizu znakova `niz2` i vraća pokazivač na znak pronađen u nizu `niz2`. U slučaju da u nizu `niz2` ne postoji niti jedan od znakova iz niza `niz1`, funkcija vraća NULL pokazivač. Npr. za zadani prvi niz nizove: `1!.` i drugi niz: `Abe!e.2!`, funkcija vraća pokazivač na 4. znak u drugom nizu (znak `!`). Napomena: nije dozvoljeno korištenje funkcija iz biblioteke `string.h`.

b) Napisati funkciju `brojRecenica` koja za zadani niz znakova vraća broj rečenica u tom nizu. Kraj rečenice označava točka, upitnik ili uskličnik. Npr. za zadani niz: "`Danas je lijep i suncan dan! Ne treba mi kisobran.`", funkcija vraća 2, a za niz: "`Pada kisa`" vraća 0, jer rečenica nije završena. Prilikom brojanja rečenica obavezno je koristiti funkciju `prviZnak`.

c) Napisati glavni program kojim će se učitati niz znakova koji sigurno nije dulji od 200 znakova. Pomoću funkcije `brojRecenica` odrediti broj rečenica u učitanom nizu i nakon toga ispisati izračunati broj rečenica, uz odgovarajuću poruku.

Npr. za zadani niz: `"Danas je lijep i suncan dan. Ne treba mi kisobran.",` ispisuje se:

```
Broj recenica u tekstu:
'Danas je lijep i suncan dan. Ne treba mi kisobran'
je 1.
```

a za niz: "`Pada kisa`", ispisuje se:

```
Tekst
'Pada kisa'
ne sadrzi niti jednu potpunu recenicu.
```

6. zadatak

a) Napisati funkciju `brojNeSlova` koja u zadanom nizu znakova određuje broj znakova koji nisu jedno od slova engleske abecede ('A' – 'Z' ili 'a' –'z').

b) Napisati funkciju `caesarEncrypt` koja temeljem ulaznog niza znakova proizvodi kriptirani niz znakova na način da svako slovo pomiče za zadani broj mjesta udesno u abecedi. Takva je enkripcija poznata pod nazivom Cezarova enkripcija, a broj mjesta za koje se obavlja posmak u abecedi se naziva ključem enkripcije. Na primjer za zadani ulazni niz 'AbZ' i ključ kriptiranja 1, Cezarovom enkripcijom se dobije znakovni niz 'Bca' dok se za isti ulazni niz i ključ kriptiranja 5 dobije znakovni niz 'FgE'. Prototip funkcije je:

```
void caesarEncrypt (char *nizUlaz, char *nizIzlaz, int kljucKriptiranja);
```

Pretpostavka je da ulazni niz znakova sadrži samo slova engleske abecede i u samoj funkciji nije potrebno provjeravati ispravnost ulaznog niza znakova.

c) U glavnom programu je potrebno s tipkovnice učitati niz znakova ne dulji od 100 znakova (nije potrebno provjeravati) kojeg se želi kriptirati Cezarovom enkripcijom. Sve dok ulazni niz znakova sadrži barem jedan znak kojeg Cezarovom enkripcijom nije moguće kriptirati, potrebno je ponavljati postupak učitavanja. Nakon toga je potrebno učitati ključ enkripcije (cijeli broj između 1 i 25) te temeljem ulaznog niza znakova proizvesti kriptirani niz. Na zaslonu ispisati originalni i kriptirani niz znakova. Obavezno koristiti funkcije iz a) i b) dijela zadatka.

Primjer ulaznih podataka i izlaznog rezultata:

```
Upisi niz: Julije Cezar
Upisi niz: JulijeCezar

Upisi kljuc kriptiranja: 27
Upisi kljuc kriptiranja: 7

Kriptiran niz: QbspqlJlghy
Originalni niz: JulijeCezar
```

7. zadatak

a) Napisati funkcije koje određuju broj riječi, odnosno broj odlomaka u tekstu,(zadanom nizu znakova). Odlomcima se smatraju nizovi znakova koji se sastoje od jedne ili više rečenica te su odvojeni znakom za novi redak ('\n'). Rečenice završavaju znakovima: '.', '!' ili '?'. Riječi su nizovi znakova koji sadrže slovo ili broj, mogu biti odvojene razmakom, zarezom ili znakovima za završetak rečenice. Prototipovi funkcija su:

```
int br_odlomaka(char * tekst);
int br_rijeci(char * tekst);
```

Podrazumijeva se da svaka rečenica i svaki odlomak u zadanom tekstu ispravno završavaju.

b) Napisati glavni program koji s tipkovnice učitava zadani tekst ne duži od 2000 znakova, poziva funkcije zadane pod a) te ispisuje broj riječi i odlomaka. Ako nema riječi u zadanom tekstu ispisuje se poruka `"Zadani tekst ne sadrzi rijeci!"`

Prilikom izvođenja programa, ulazne podatke preuzeti iz datoteke **`tekst.txt`**, umjesto s tipkovnice (koristiti preusmjeravanje iz datoteke).

Primjer zadanog tekst:

```
Naranca je suptropska biljka iz porodice Rutaceae. Moze zivjeti do 500 godina.
Plod je ukusan i socan, ovalnog oblika. Zimzeleno je drvo iz jugoistocne Azije,
iz roda Citrus. Uzgaja se u regijama s toplom klimom poput Sredozemlja, Juzne
Afrike i Kalifornije. U zemlji podrijetla Indiji, na sanskrtu se naziva na rangi
ili naranja, dok su Englezi prijevod norange pretvorili u današnji orange.  U
dosta jezika naranca se naziva "kineska jabuka" (npr. u nizozemskom
"Sinaasappel", te u njemackom "Apfelsine"). Po rijeci naranca nazvana je
narancasta boja.
Rijec naranca dolazi iz sanskritske rijeci narangaḥ, što znači narancino drvo.
Macke odbija kora narance.
```

Za primjer zadanog teksta program će ispisati:

```
Zadani tekst sadrzi rijeci/odlomaka: 100/2.
```

8. zadatak

a) Napisati funkciju `brLijevihiDesnihGranicnika` koja u zadanom nizu znakova određuje broj pojavljivanja znakova `lijevi` i `desni` koji su joj predani kao argumenti. Prototip funkcije je:

```
void brLijevihiDesnihGranicnika (char *znNiz, char lijevi, char desni
                                 , int *brLijevih, int *brDesnih);
```

b) Napisati funkciju `izrazJeIspravan` koja određuje je li zadani niz znakova ispravan obzirom na proslijeđena dva znaka: lijevi i desni graničnik. Niz znakova je ispravan ako sadrži jednak broj lijevih i desnih graničnika. Za ispravan niz znakova funkcija vraća vrijednost 1, inače vraća vrijednost 0.

c) U glavnom programu je potrebno s tipkovnice učitati niz znakova ne dulji od 200 znakova (nije potrebno provjeravati) koji predstavlja izraz kojeg se želi evaluirati. Potom je potrebno učitavati dva znaka koji predstavljaju par graničnika (lijevi i desni). Za svaki učitani par graničnika, potrebno je odrediti je li učitani niz znakova ispravan obzirom na zadani par graničnika i ispisati odgovarajuću poruku. Niz znakova je ispravan ako sadrži jednak broj lijevih i desnih graničnika pri čemu (radi jednostavnosti) nije važan redoslijed njihovog pojavljivanja (npr. izraz „(a+b)" je ispravan obzirom na graničnike „(" i „)" ali i izraz ")a+b(" je također ispravan obzirom na iste graničnike).

Učitavanje graničnika i evaluaciju izraza je potrebno prekinuti kada se za bilo koji (ili oba) graničnika unese znak „\". Na kraju je potrebno ispisati poruku koja će izraz kvalificirati bilo kao potpuno ispravan obzirom na sve provjeravane graničnike, kao potpuno neispravan obzirom na sve provjeravane graničnike ili kao djelomično ispravan. Tekst poruke formulirati po volji. Obavezno koristiti funkcije iz a) i b) dijela zadatka.

Primjer ulaznih podataka i izlaznog rezultata:

```
Upisite izraz koji zelite provjeriti: {[(x1+x2)*5 -)5-x1(%3

Upisite lijevi i desni granicnik: []
Izraz {[(x1+x2)*5 -(5-x1)%3  NIJE ispravan obzirom na granicnike [ ].

Upisite lijevi i desni granicnik: {}
Izraz {[(x1+x2)*5 -(5-x1)%3  NIJE ispravan obzirom na granicnike { }.

Upisite lijevi i desni granicnik: ()
Izraz {[(x1+x2)*5 -(5-x1)%3  JE ispravan obzirom na granicnike ( ).

Upisite lijevi i desni granicnik: \ \

Izraz {[(x1+x2)*5 -(5-x1)%3 je djelomicno ispravan: ispravan obzirom na 1
granicnik(a); neispravan obzirom na 2 granicnik(a).
```

9. zadatak

a) Napisati funkciju `genPodniz` čiji je prototip:

```
void genPodniz(char *nizUlaz, char *podNiz, int pocPozicija, int duljPodniz);
```

koja će na temelju ulaznog niza znakova (`nizUlaz`) generirati podniz zadane duljine (`duljPodniz`) počevši od zadane pozicije (`pocPozicija`). Na primjer, za ulazni niz znakova `"Posao"` i zadanu `duljPodniz` jednaku 3 te početnu poziciju `pocPozicija` jednaku 0, funkcija niz znakova `podNiz` treba napuniti sadržajem `"Pos"`, dok za početnu poziciju jednaku 1, njegov sadržaj treba biti `"osa"` itd.

b) Napisati funkciju `nizoviJednaki` čiji je prototip:

```
int nizoviJednaki (char *niz1, char *niz2);
```

koja će vratiti vrijednost 1 ako su dva ulazna niza znakova jednaka. U protivnom vraća vrijednost 0. U ovoj funkciji nije dozvoljeno koristiti funkcije za usporedbu nizova znakova iz `string.h` biblioteke.

c) Glavnim programom je potrebno odrediti i ispisati sličnost između dva niza znakova koja se odredi temeljem broja podudarnih podnizova zadane duljine. Potrebno je s tipkovnice učitati dva niza znakova ne dulja od 300 znakova (nije potrebno provjeravati) i cijeli broj (ne veći od 5 niti manji od 2) koji predstavlja duljinu podnizova. Sličnost između učitanih nizova je definirana kao omjer broja zajedničkih podnizova (koji se nalaze u oba niza) pomnoženog s 2 i ukupnog broja podnizova.

Na primjer za ulazne znakovne nizove `"Posao"` i `"Pisao"`, i učitanu duljinu podniza jednaku 3, mogući podnizovi su `{"Pos", "osa", "sao"}` odnosno `{"Pis", "isa", "sao"}`. Broj zajedničkih podnizova je 1 `{"sao"}` a ukupan broj podnizova je 3+3=6 `{"Pos", "osa", "sao", "Pis", "isa", "sao"}`. Sličnost iznosi 1*2/(3+3)=0,333333. Opisani postupak je pojednostavljena verzija jedne od metoda koje se koriste pri približnoj pretrazi teksta (Fuzzy String Matching).

Obavezno koristiti funkcije iz a) i b) dijela zadatka.

Primjer ulaznih podataka i izlaznog rezultata:

```
Upisi prvi niz:Posao
Upisi drugi niz:Pisao
Na podskupove od koliko znakova zelis rastavljati nizove:3

Slicnost nizova Posao i Pisao iznosi 0.333333
```

10. zadatak

a) Napisati funkciju `brojPonavljanja` koja u zadanome nizu znakova vraća broj uzastopnih ponavljanja zadanog znaka prilikom njegovog prvog pojavljivanja u zadanom nizu. Npr. za niz `aaabbcddddbbb` i zadani znak `d`, funkcija će vratiti broj 4, za zadani znak `b` funkcija će vratiti broj 2, a za zadani znak `e` funkcija će vratiti 0. Prototip funkcije je:

```
int brojPonavljanja(char niz[], char znak);
```

b) Napisati funkciju `kodiraj` koja zadani niz znakova kodira tako da uzastopna ponavljanja jednog znaka zamjenjuje tim znakom i brojem pojavljivanja tog znaka. Npr. niz `aaabbcddddbbb` zamijenit će se nizom `a3b2c1d4b3`. Može se pretpostaviti da se pojedino slovo neće uzastopno pojaviti više od 9 puta. Za određivanje broja uzastopnog pojavljivanja određenog znaka potrebno je koristiti funkciju iz a) dijela zadatka. Prototip funkcije je:

```
void kodiraj(char src[], char dest[]);
```

c) Napisati glavni program u kojem je potrebno učitati znakovni niz ne dulji od 100 znakova (nije potrebno provjeravati), kodirati učitani niz funkcijom `kodiraj` i ispisati originalni i kodirani niz.

11. zadatak

a) Napisati funkciju `nadjiPodniz` koja u zadanom nizu znakova traži zadani podniz i vraća pokazivač na prvi znak podniza u znakovnom nizu. Ako zadani podniz nije pronađen, funkcija vraća `NULL` pokazivač. Npr. za niz znakova `aaabaccdddd` i podniz `bac`, funkcija vraća pokazivač na 4. znak u zadanom nizu znakova (znak `b`). Napomena: nije dozvoljeno korištenje funkcija iz biblioteke `string.h`.

Funkcija ima prototip:

```
char* nadjiPodniz(char *ulaz, char *uzorak);
```

b) Napisati funkciju `izbaci` koja iz zadanog niza znakova izbacuje prvo pojavljivanje zadanog podniza. Npr. za niz znakova `aaabaccdddd` i podniz `bac`, funkcija će izmijeniti niz u `aaacdddd`. Ako je podniz pronađen i izbačen funkcija vraća 1, a inače 0. Funkcija ima prototip:

```
int izbaci(char *ulaz, char *uzorak);
```

Napomena: za određivanje pozicije podniza koristiti funkciju iz a) dijela zadatka.

c) Napisati glavni program u kojem je potrebno učitati niz znakova (`ulaz`) ne dulji od 100 znakova, a zatim učitati drugi niz znakova (`uzorak`) ne dulji od 10 znakova (nije potrebno provjeravati duljinu učitanih nizova). Koristeći funkciju iz b) dijela zadatka, u učitanom nizu znakova `ulaz`, potrebno je izbaciti sva pojavljivanja znakovnog niza `uzorak`.

Primjer:

Sadržaj niza `ulaz` je: `aaababaccddddbbac`
Sadržaj niza `uzorak` je: `bac`
Nakon izvođenja glavnog programa sadržaj niza `ulaz` je: `aaaddddb`

12. zadatak

Dva najčešća slova u standardnom tekstu hrvatskog jezika su A i I, u engleskom jeziku to su redom E i T, dok su u njemačkom redom E i N. Za zadani tekst (ne duži od 1500 znakova) pohranjen u datoteci **`tekst.txt`** potrebno je obaviti jednostavnu analizu učestalosti pojavljivanja pojedinih znakova.

a) Napisati funkciju koja za zadani niz znakova pronalazi dva znaka koji se najčešće u njemu pojavljuju (pretpostaviti da se koriste samo slova engleske abecede).

b) Napisati glavni program koji s tipkovnice učitava niz znakova (**`tekst`**), ne duži od 1500 znakova, te ovisno o učestalosti pojavljivanja karakterističnih slova „pokušava odgonetnuti“ na kojem jeziku je napisan zadani tekst.

   Ovisno o rezultatu „detekcije“ potrebno je ispisati poruku sljedećeg sadržaja:

   ```
   Jezik na kojem je tekst vjerojatno napisan je:
   ```

   i jedan od sljedećih tekstova:

   ```
   Hrvatski / Engleski / Njemački / Nepoznat
   ```

   Ako je najčešće slovo A ili I, pretpostaviti da je tekst na hrvatskom, ako je najčešće slovo E, potrebno je analizirati “drugo najčešće” slovo kako bi se odlučilo da li je riječ o engleskom ili njemačkom jeziku. U svim ostalim slučajevima, jezik je “Nepoznat”.

   Prilikom izvođenja programa, ulazne podatke preuzeti iz datoteke **`tekst.txt`**, umjesto s tipkovnice (koristiti preusmjeravanje iz datoteke).

Primjeri tekstova i ispisa programa:

```
Limun je jedna od najkorisnijih namirnica na nasem planetu. Stiti od jako
velikog broja bolesti, tako da ga se s pravom moze nazvati super-hranom. Pun je
nutrijenata: sadrzi bioflavonoide, pektin, folnu kiselinu, vitamini C, A, B1,
B6, kalij, kalcij, magnezij, fosfor, mangan. Povoljno djeluje na rad jetre,
crijeva, zeluca, imunog, zivcanog i kardiovaskularnog sustava.
```

`Jezik na kojem je tekst vjerojatno napisan je: Hrvatski`

```
The origin of kebab may lie in the short supply of cooking fuel in the Near
East; tradition has it that the dish was invented by medieval Turkic or Persian
soldiers who used their swords to grill meat over open-field fires. The dish has
been native to the Near East.
```

`Jezik na kojem je tekst vjerojatno napisan je: Engleski`

```
La llama es un mamífero artiodactilo domestico de la familia Camelidae,
abundante en la Puna o Altiplano de los Andes de Argentina, Bolivia, Chile,
Ecuador y Peru. Fue creado por los pueblos andinos nativos mediante seleccion
artificial a partir de guanacos salvajes que fueron domesticados, del cual, por
lo tanto, la llama deriva. La llama sigue siendo utilizada por los pobladores
andinos por su lana, carne, y como transporte de mercancia.
```

`Jezik na kojem je tekst vjerojatno napisan je: Nepoznat`

13. zadatak

a) Potrebno je napisati funkciju čiji je prototip:

```
int intUNiz(int broj, char *niz);
```

Funkcija znamenke zadanog cijelog broja iz intervala [0, 59] pohranjuje u niz znakova `niz`. Ako je zadani broj jednoznamenkast, onda se na mjestu desetica u nizu znakova postavlja znamenka '0'. Ako je zadani cijeli broj iz intervala [0, 59], funkcija vraća vrijednost 1, a 0 inače.

Primjer:
Za zadani cijeli broj 12, generiran je niz znakova "12", a funkcija vraća vrijednost 1.
Za zadani cijeli broj 2, generiran je niz znakova "02", a funkcija vraća vrijednost 1.
Za zadani cijeli broj -2, funkcija vraća vrijednost 0.

b) Potrebno je napisati funkciju čiji je prototip:

```
char *dodajNiz(char *prvi, char *drugi);
```

u kojoj se niz `drugi` dodaje na kraj niza `prvi`. Pretpostaviti da nakon niza `prvi` postoji dovoljno mjesta za dodavanje niza `drugi`. Funkcija treba vratiti pokazivač na početak niza `prvi`.

c) Potrebno je napisati funkciju koja zadani cijeli broj `sekunde` preračunava u sate, minute i sekunde (argumenti `h`, `m`, `s`). Prototip funkcije je:

```
void hms(int sekunde, int *h, int *m, int *s);
```

d) U glavnom programu potrebno je pomoću tipkovnice unijeti broj koji predstavlja vrijeme u sekundama. Učitano vrijeme koje predstavlja broj sekunda pretvoriti u sate, minute i sekunde te izračunate sate, minute i sekunde pretvoriti u niz znakova oblika hh:mm:ss korištenjem funkcije `intUNiz` i `dodajNiz`.

Primjer:
Za ulazni broj sekundi 3605 treba ispisati niz:
`01:00:05`

Za ulazni broj sekundi 5 treba ispisati niz:
`00:00:05`

14. zadatak

a) Potrebno je napisati funkciju `nizUInt` koja pretvara ispravno zadan ulazni niz u cijeli nenegativni broj, koji vraća u pozivajući program. Ulazni niz znakova je ispravno zadan, ako se sastoji od dekadskih znamenki. Ako ulazni niz znakova nije ispravno zadan, funkcija vraća vrijednost -1.

Primjer:

Za ulazni niz "`1234`" funkcija treba vratiti broj `1234`.
Za ulazni niz "-`2345`" funkcija treba vratiti `-1`.
Za ulazni niz "`02345`" funkcija treba vratiti `2345`.
Za ulazni niz "`a12`" funkcija treba vratiti `-1`.

b) Potrebno je napisati funkciju `matemOp` u kojoj se za dva zadana cijela broja i zadani matematički operator izračunava rezultat matematičke operacije zadane ulaznim operatorom. Matematička operacija može biti samo zbrajanje ili oduzimanje, a zadaje se znakom '+' ili '-'. Ako je matematički operator ispravno zadan, funkcija vraća vrijednost 1, a u suprotnom vraća 0. Rezultat matematičke operacije treba prenijeti preko argumenta funkcije u pozivajući program.

c) U glavnom programu potrebno je pomoću tipkovnice unijeti dva niza znakova, ne dulja od 9 znakova, te znak koji predstavlja matematičku operaciju. Ako su oba operanda i operator ispravno zadani, na zaslon je potrebno ispisati rezultat, a inače je potrebno dojaviti odgovarajuću poruku o pogrešci. Koristiti funkcije `nizUInt` i `matemOp`.

Primjer:

Za zadane ulazne nizove "`123`", "`100`" i znak '+', potrebno je na zaslon ispisati:

```
Rezultat je 223.
```

Za zadane ulazne nizove "`123`", "`01a`" i znak '+', potrebno je na zaslon ispisati:

```
Učitani niz "01a" nije ispravno zadan.
```

Za zadane ulazne nizove "`123`", "`10`" i znak '!', potrebno je na zaslon ispisati:

```
Operator nije ispravno zadan.
```

15. zadatak

a) Napisati funkciju čiji je prototip:

```
int sazetak(char *ulaz);
```

Funkcija treba izračunati i vratiti brojevni sažetak niza znakova zadanih argumentom `ulaz`, koji je definiran kao suma ASCII kodova svih ulaznih znakova modulo 128. Dakle, za niz "`Niz`" sažetak je (78 + 105 + 122) mod 128 = 49 i funkcija vraća 49. Za zadani prazan niz ili NULL funkcija vraća broj 0.

b) Napisati funkciju zadanu prototipom:

```
void dodajZnak(char *ulaz, int broj);
```

koja na kraj ulaznog niza (`ulaz`) dodaje znak čija je ASCII vrijednost zadana argumentom `broj`. Pretpostaviti da u nizu `ulaz` ima dovoljno mjesta za dodavanje još jednog znaka. Primjer: ako je zadan ulazni niz "`ovo`" i broj 84, onda je izmijenjeni niz: "`ovoT`".

c) U glavnom programu potrebno je učitati niz znakova ne dulji od 200 znakova. Učitani niz znakova predstavlja rečenicu koja se sastoji od riječi. Pretpostaviti da je riječ niz malih i velikih slova engleske abecede. Riječi u rečenici su međusobno odvojene prazninom. Nije potrebno provjeravati je li rečenica (ulazni niz znakova) ispravno zadana.

Za svaku riječ u rečenici potrebno je odrediti njezin brojevni sažetak, a zatim ispisati riječ, njezin brojevni sažetak i pripadajuću proširenu riječ, tj. riječ kojoj je na kraju dodan njezin brojevni sažetak. Obavezno koristiti funkcije iz a) i b) dijela zadatka.

Primjer:
Za ulazni niz "ovo niz rijeci" program treba ispisati:

```
ovo 84 ovoT
niz 81 nizQ
rijeci 118 rijeciv
```

16. zadatak

a) Napisati funkciju `minMax` koja u zadanom dvodimenzionalnom cjelobrojnom polju pronalazi najmanju i najveću vrijednost elemenata polja. Npr: za 1. polje funkcija u pozivajući program "vraća" -5 i 9, za 2. polje vrijednosti -9 i 11, a za 3. polje vrijednosti 1 i 1:

```
 1 -5
-2  3
 8 -5
 4  9
```

```
  1  -1   2  10  11
  2   3  -2  -5   3
 11  -2   3   3  -9
```

```
1  1
1  1
```

b) Napisati funkciju `dobreDimenzije` koja vraća vrijednost 1 (cijeli broj), ako su dimenzije dvodimenzionalnog polja (broj redaka i broj stupaca) ispravne s obzirom na definiranu maksimalnu veličinu polja, a u suprotnom vraća vrijednost 0. Stvarne i maksimalne dimenzije polja su argumenti funkcije. Funkcija mora raditi i za polja kod kojih maksimalni broj redaka nije jednak maksimalnom broju stupaca.

c) U glavnom programu potrebno je s tipkovnice učitati dimenziju (broj redaka i broj stupaca) dvodimenzionalnog polja cijelih brojeva maksimalne dimenzije 20x20. Broj redaka i broj stupaca učitava se sve dok se ne učitaju ispravna dimenzije (s obzirom na definiranu maksimalnu dimenziju polja). Nakon toga učitati elemente polja, odrediti najmanju i najveću vrijednost elemenata polja te na zaslon, u obliku matrice, ispisati elemente polja te najmanju i najveću vrijednost. Obavezno koristiti funkcije `minMax` i `dobreDimenzije`.

17. zadatak

a) Napisati funkciju `kvadrat` koja svaki element zadanog dvodimenzionalnog cjelobrojnog polja zamjenjuje s kvadratom njegove vrijednosti.

b) Napisati funkciju `minStupac` koja za zadano dvodimenzionalno cjelobrojno polje (matricu) određuje najmanji element za svaki stupac. Funkcija minimume pohranjuje u jednodimenzionalno polje (argument funkcije), gdje indeksi elementa polja odgovaraju indeksu stupca. Npr. najmanji element u stupcu matrice s indeksom 0 treba biti pohranjen kao član s indeksom 0 u jednodimenzionalnom polju, itd.

c) U glavnom programu je potrebno s tipkovnice učitati broj redaka i broj stupaca dvodimenzionalnog polja (matrice) cijelih brojeva maksimalnih dimenzija 10 x 10. Broj redaka i broj stupaca je potrebno učitavati sve dok se ne učita ispravan broj redaka i stupaca s obzirom na definirane maksimalne vrijednosti. Svaki element matrice potrebno je zamijeniti kvadratom njegove vrijednosti i nakon toga ispisati matricu s promijenjenim vrijednostima elemenata te za svaki stupac njegov najmanji element. Najmanji je član pojedinog stupca matrice potrebno ispisati kao zadnju vrijednost u pripadajućem stupcu (vidjeti primjer). Obavezno koristiti funkcije iz a) i b) dijela zadatka.

Primjer: Ako je zadana ulazna matrica:

```
   7  -1   2  10
   2   4  -2  -8
  10  -4   3   3
```

na zaslon treba ispisati:

```
  49   1   4 100
   4  16   4  64
 100  16   9   9
   4   1   4   9
```

18. zadatak

a) Napisati funkciju `prosjek` koja za zadano dvodimenzionalno cjelobrojno polje, računa prosjek po pojedinom retku te prosjek po pojedinom stupcu. Funkcija prosjeke pohranjuje u jednodimenzionalna polja (argumenti funkcije), gdje indeksi elementa polja odgovaraju indeksu retka odnosno stupca.

b) Napisati glavni program koji s tipkovnice učitava dimenziju (ne veću od 10) kvadratnog dvodimenzionalnog polja cijelih brojeva, provjerava učitanu dimenziju polja te ponavlja učitavanje do unosa ispravne vrijednost, a zatim učitava članove polja. Nakon toga ispisuje elemente polja i izračunate prosjeke: u prvom stupcu program ispisuje prosjeke po retcima, a u zadnjem retku prosjeke po stupcima. Obavezno koristiti funkciju iz a) dijela zadatka.

Primjer ulaznih podataka i izlaznog rezultata:

```
Unesite ispravnu dimenziju polja: 3
Unesite elemente polja:
p[0][0]: 2
p[0][1]: 2
p[0][2]: 3
p[1][0]: 4
p[1][1]: 5
p[1][2]: 6
p[2][0]: 7
p[2][1]: 8
p[2][2]: 9
Matrica:

2.33       2       2       3
5.00       4       5       6
8.00       7       8       9
        4.33    5.00    6.00
```

19. zadatak

a) Napisati funkciju čiji je prototip:

```
void rastavi(char *niz, int brojRijeci, int maxRijeci,
             int duljinaRijeci, char *polje);
```

Funkcija zadani niz znakova `niz` rastavlja na riječi i u dvodimenzionalno polje znakova `polje` upisuje najviše prvih `brojRijeci` riječi iz zadanog niza. U svakom retku dvodimenzionalnog polja mora biti pohranjena jedna riječ, a kraj riječi određen je znakom '\0'. Redak polja u kojem se ne nalazi riječ sadrži prazan niz znakova. U dvodimenzionalno polje moguće je pohraniti najviše `maxRijeci` riječi, a pojedina riječ ne može imati više od `duljinaRijeci` znakova (ako u nizu postoji dulja riječ, pohranjuje se prvih `duljinaRijeci` znakova). Riječi su u zadanom nizu međusobno odvojene jednom prazninom (znak *blank*).

Npr. za zadani niz: "`Danas je lijep i suncan dan`", broj riječi 5 i najveću duljinu riječi 6, sadržaj polja u kojem može biti najviše 5 riječi je:

'D'	'a'	'n'	'a'	's'	'\0'	'\0'
'j'	'e'	'\0'	'\0'	'\0'	'\0'	'\0'
'l'	'i'	'j'	'e'	'p'	'\0'	'\0'
'i'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'
's'	'u'	'n'	'c'	'a'	'n'	"'\0"

, a za niz: "`Imam kisobran`", broj riječi 3 i duljinu riječi 4, sadržaj polja u kojem može biti najviše 4 riječi je:

'I'	'm'	'a'	'm'	'\0'
'k'	'i'	's'	'o'	'\0'
'\0'	'\0'	'\0'	'\0'	'\0'
'\0'	'\0'	'\0'	'\0'	'\0'

b) Napisati funkciju `brojRijeci` koja računa broj riječi u zadanom dvodimenzionalnom polju znakova u kojem u je svakom retku pohranjena jedna riječ, a kraj riječi određen je znakom '\0'. Redak polja u kojem se ne nalazi riječ sadrži prazan niz. Npr. u prvom primjeru iz a) dijela zadatka funkcija vraća 5, a u drugom primjeru vraća 2.

c) Napisati glavni program kojim će se učitati niz znakova koji sigurno nije dulji od 200 znakova i broj riječi koje je potrebno upisati u dvodimenzionalno polje (učitane vrijednosti nije potrebno provjeravati). U dvodimenzionalno polje je moguće upisati najviše 100 riječi, a pojedina riječ ne smije biti dulja od 20 znakova. Nakon učitavanja potrebnih vrijednosti, funkcijom `rastavi` popuniti dvodimenzionalno polje znakova. Na zaslon ispisati broj riječi upisanih u polje, dobiven pomoću funkcije `brojRijeci`, a zatim i riječi pohranjene u polju (svaku riječ ispisati u zasebnom retku).