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 realni 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:

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

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

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* $\in$ [-180, 180], specificira sjever-jug i *longitude* $\in$ [-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: 23000 44.114934 15.228952
Upisite postanski broj i koordinate 3. grada: 21000 43.50692 16.44245

Medjusobne udaljenosti gradova:
10000 - 23000: 197.924 km
10000 - 21000: 259.122 km
23000 - 21000: 118.510 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 (*unsigned*) 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. Prototip funkcije je:

```
void unija(int n1, int *s1, int n2, int *s2, int *nrez, int *rez);
```

b) Budući da u programskom jeziku C ne postoji tip podatka "skup", potrebno je skup ostvariti poljem, pri čemu treba paziti na svojstvo skupova – da ne mogu sadržavati duplikate. 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). Ako se učita duplikat, ne uključiti ga u skup! Pretpostaviti da ulazni skupovi neće imati više od 20 elemenata. Program, nakon učitavanja elemenata skupova, ispisuje sadržaj svakog skupa, te pomoću funkcije iz a) dijela zadatka izračunava uniju, te ju ispisuje na zaslon.

Npr. Ako korisnik unese:
```
Skup 1: 2,8,4,1,6,7,8,2,5
Skup 2: 5,1,2
```

Program treba ispisati:
```
Skup 1: 2,8,4,1,6,7,5
Skup 2: 5,1,2
Unija: 2,8,4,1,6,7,5
```

, a za unos:
```
Skup 1: 2,8,4,1,6,7,8
Skup 2: 5,1
```

Program treba ispisati:
```
Skup 1: 2,8,4,1,6,7
Skup 2: 5,1
Unija: 2,8,4,1,6,7,5
```

4. zadatak

a) Napisati funkciju čiji je prototip:

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

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

- 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`.
- 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`.

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: `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 2.
```

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

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

6. zadatak

a) Napisati funkciju koja određuje broj riječi u zadanom tekstu. Riječi su nizovi znakova koji sadrže slovo engleske abecede ili brojeve (znamenke 0-9), a odvojene su razmakom, zarezom ili znakovima za završetak rečenice. Tekst koji sadrži bilo koji drugi znak smatra se neispravnim i funkcija u tom slučaju vraća vrijednost -1. Prototip funkcije je:

```
int brRijeci(char * tekst);
```

b) Napisati funkciju koja iz zadanog teksta izbacuje sve brojeve, npr. zadani tekst „Ponovljeno je 100 puta“ funkcija mijenja u „Ponovljeno je puta.“. Ako je tekst promijenjen, funkcija vraća 1, inače 0. Prototip funkcije je:

```
int izbaciBrojeve(char * tekst);
```

c) Napisati glavni program koji s tipkovnice učita tekst ne duži od 1024 znakova. Program prvo ispisuje broj riječi u učitanom tekstu, a zatim iz učitanog teksta izbacuje sve brojeve. Za izračunavanje broja riječi u tekstu i izbacivanje brojeva iz teksta koriste se funkcije iz a) i b) dijela zadatka. Ako je tekst promijenjen, program ispisuje promijenjeni tekst i broj riječi u promijenjenom tekstu.

Primjer 1: Za zadani tekst
```
Naranca je suptropska biljka iz porod67ice Rutaceae. Moze zivjeti do 500 godina.
```

program će ispisati:
```
Broj rijeci u zadanom tekstu: 12

Promijenjen tekst:
Naranca je suptropska biljka iz porodice Rutaceae. Moze zivjeti do  godina.
Broj rijeci poslije promjene je: 11
```

Primjer 2: Za zadani tekst
```
Narancajesuptropskabiljka
```

program će ispisati:
```
Broj rijeci u zadanom tekstu: 1
```

Primjer 3: Za zadani tekst
```
Naranca je#suptropsk a&&biljka
```

program će ispisati:
```
Zadani tekst je neispravan.
```

Primjer 4: Za zadani tekst
```
Naranca je suptropska biljka iz porodice Rutaceae.
```

program će ispisati:
```
Broj rijeci u zadanom tekstu: 7
```

7. 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
```

8. zadatak

a) Napisati funkciju `brojacGranicnihZnakova` koja u zadanom nizu znakova `znNiz` određuje broj pojavljivanja dva znaka `lijevi` i `desni`. Prototip funkcije:

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

b) Napisati funkciju `izrazJeIspravan` koja vraća vrijednost 1 ako zadani ulazni niz znakova sadrži jednak broj dvaju proslijeđenih znakova (`lijevi` i `desni`), inače vraća vrijednost 0. Prototip funkcije:

```
int izrazJeIspravan(char *znNiz, char lijevi, char desni);
```

c) U glavnom programu je potrebno s tipkovnice učitati niz znakova ne dulji od 200 (nije potrebno provjeravati). Potom je potrebno učitavati po dva znaka (lijevi i desni granični znak) i provjeravati je li ulazni niz ispravan obzirom na zadane granične znakove. Odgovarajuću poruku ispisati za svaki par graničnih znakova. Niz znakova je ispravan ako sadrži jednak broj lijevih i desnih graničnih znakova pri čemu (radi jednostavnosti) nije važan redoslijed njihovog pojavljivanja (npr. izraz ”(a+b)” je ispravan s obzirom na par graničnih znakova ”(“ i ”)“ ali i izraz ”)a+b(” je također ispravan s obzirom na isti par graničnih znakova).

Učitavanje para graničnih znakova i evaluaciju izraza je potrebno prekinuti kada se za bilo koji (ili oba) granična znaka unese znak “\“.

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: \.
```

9. zadatak

a) Napisati funkciju `genPodniz` čiji je prototip:

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

koja će na temelju ulaznog niza znakova (`niz`) generirati podniz zadane duljine (`duljPodniz`) počevši od zadane pozicije (`pocPozicija`). Funkcija ne treba provjeravati jesu li ulazni parametri ispravni.

Primjer: za ulazni niz znakova `"cvrci cvrcak"` i zadanu `duljPodniz` jednaku 4 te početnu poziciju `pocPozicija` jednaku 0, funkcija niz znakova `podNiz` treba napuniti sadržajem `"cvrc"`, a za početnu poziciju jednaku 1, sadržajem `"vrci"` itd.

b) Napisati funkciju `sadrziPodniz` čiji je prototip:

```
int sadrziPodniz(char *niz, char *podNiz);
```

koja vraća vrijednost 1 ako je podniz (`podNiz`) sadržan u nizu (`niz`), a u suprotnom vraća vrijednost 0. U ovoj funkciji nije dozvoljeno koristiti funkcije iz biblioteke `string.h`.

c) Glavnim programom je potrebno odrediti i ispisati sličnost između dva niza znakova. 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 podniza. Duljinu podniza potrebno je učitavati sve dok se ne učita ispravna vrijednost - ona koja je u zadanom intervalu i čija je duljina manja ili jednaka duljini svakog od ulaznih nizova.

Sličnost prvog niza s drugim je definirana kao omjer broja podnizova prvog niza koji su sadržani u drugom nizu i broja podnizova u prvom nizu.

Na primjer za ulazne nizove `"cvrci cvrcak"` i `"cvrcanje cvrcka"`, i učitanu duljinu podniza jednaku 4, provjerava se koji su podnizovi prvog niza:
`{"cvrc", "vrci", "rci ", "ci c", "i cv", " cvr", "cvrc", "vrca", "rcak"}`
sadržani u drugom nizu.

Broj podnizova prvog niza je 9, a broj podnizova prvog niza koji su sadržani u drugom nizu je 4 `{"cvrc", " cvr", "cvrc", "vrca"}`. Sličnost, dakle iznosi 4/9=0,4444.

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:

```
Prvi niz : knjizevnost i kazaliste
Drugi niz: knjizevna vrsta

Duljina podniza za usporedbu nizova: 5

Slicnost nizova 'knjizevnost i kazaliste' i 'knjizevna vrsta' iznosi 0.210526
```

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

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

b) Dva najčešća slova u standardnom tekstu hrvatskog jezika su A i I (neovisno o redoslijedu), u engleskom jeziku to su redom E pa T, dok su u njemačkom redom E pa N. 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 (neovisno o redoslijedu), 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”.

Primjeri tekstova i ispisa programa:

```
Ova recenica je slicnih svojstava kao i vecina drugih iz jezicnog korpusa.
```

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

```
Ein einfacher Satz geschrieben in Deutsch.
```

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

```
Secret agent never misses the target
```

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

```
In hoc signo vinces.
```

`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 za ispravno zadani ulazni niz znakova (*string*) u pozivajući program vraća odgovarajući cijeli nenegativni broj. Ulazni niz znakova je ispravno zadan ako sadrži isključivo znakove koji predstavljaju dekadske znamenke. Ako ulazni niz znakova nije ispravno zadan, funkcija `nizUInt` vraća vrijednost -1.

Primjeri:

Za ulazni niz "1234" funkcija treba vratiti cijeli broj 1234.

Za ulazni niz "-2345" funkcija treba vratiti cijeli broj -1.

Za ulazni niz "02345" funkcija treba vratiti cijeli broj 2345.

Za ulazni niz "a12" funkcija treba vratiti cijeli broj -1.

b) Potrebno je napisati funkciju `dijeljenje` koja za ulazne cjelobrojne argumente m i n izračunava rezultat dijeljenja m:n, kao realni broj jednostruke preciznosti.

c) U glavnom programu potrebno je pomoću tipkovnice unijeti dva niza znakova, ne dulja od 9 znakova. Ako se nizovi mogu pretvoriti u nenegativne cijele brojeve m i n, te vrijednost za n nije nula, izračunati i ispisati rezultat dijeljenja m:n. Inače, ispisati jednu od odgovarajućih poruka (vidjeti primjere). Koristiti funkcije `nizUInt` i `dijeljenje`.

Primjeri:

Za zadane ulazne nizove "123" i "1000", potrebno je na zaslon ispisati:
```
123 : 1000 = 0.123000
```

Za zadane ulazne nizove "12345678" i "3", potrebno je na zaslon ispisati:
```
12345678 : 3 = 4115226.000000
```

Za zadane ulazne nizove "123" i "01a", potrebno je na zaslon ispisati:
```
Niz "01a" je neispravan.Prekidam.
```

Za zadane ulazne nizove "-123" i "01a", potrebno je na zaslon ispisati:
```
Niz "-123" je neispravan.Niz "01a" je neispravan.Prekidam.
```

Za zadane ulazne nizove "123" i "0", potrebno je na zaslon ispisati:
```
Nazivnik je nula.Prekidam.
```

Za zadane ulazne nizove "-123" i "0", potrebno je na zaslon ispisati:
```
Niz "-123" je neispravan.Nazivnik je nula.Prekidam.
```

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 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

U nekoj igri, koja se sastoji od više krugova, unaprijed se dogovori broj krugova. Igru započinju dva ili više igrača, a u svakom krugu igre jedan igrač zaradi negativan bod. Igrač koji sakupi dva negativna boda više od broja negativnih bodova bilo kojeg od preostalih igrača, ispada iz igre (tj. ne sudjeluje u narednim krugovima igre). Igra završava kada se odigra unaprijed dogovoreni broj krugova ili kada u igri ostane samo jedan igrač.

a) Napisati funkciju `simulirajIgru` koja za zadani broj igrača i broj krugova igre simulira tijek cijele igre. U funkciji se za svaki krug igre, pomoću generatora slučajnih brojeva, odabire igrač koji u tom krugu dobiva negativan bod.
b) Napisati glavni program kojim se pomoću tipkovnice zada broj igrača (najviše 10) i broj krugova (najviše 100) - učitane vrijednosti nije potrebno provjeravati. Nakon učitavanja potrebnih vrijednosti, jednim pozivom funkcije `simulirajIgru` simulirati igru, a nakon toga na zaslon ispisati informacije o odigranoj igri na sljedeći način:

```
igrac   negativnih bodova
  1       4 - ispao u 8. krugu
  2       2
  3       2 - ispao u 3. krugu
```

Prikazane informacije odgovaraju igri koju su igrala 3 igrača u najviše 10 krugova, a u pojedinim krugovima su negativne bodove dobivali sljedeći igrači:

```
Krug:   1  2  3  4  5  6  7  8
Igrač:  1  3  3  2  1  2  1  1
```

18. zadatak

a) Hammingova udaljenost između nizova *s* i *t* je broj mjesta na kojima se znakovi u nizovima razlikuju, tj. broj mjesta za koja vrijedi vrijedi $s[i] \neq t[i]$.
Potrebno je napisati funkciju `HammingovaUdaljenost` koja za dva znakovna niza *s* i *t* jednake duljine određuje i vraća cijeli broj koji predstavlja broj mjesta na kojima se znakovi u nizovima *s* i *t* razlikuju.

Primjer:
Za ulazne nizove "barka" i "sarma" funkcija `HammingovaUdaljenost` treba vratiti 2.
Za ulazne nizove "barka" i "barka" funkcija `HammingovaUdaljenost` treba vratiti 0.
Za ulazne nizove "0001012" i "1101002" funkcija `HammingovaUdaljenost` treba vratiti 3.

b) Potrebno je napisati funkciju `odrediSlucajniNiz`:

```
void odrediSlucajniNiz(int n, char *s);
```

koja kao argumente prima prirodni broj *n* (duljinu niza) i pokazivač *s* na početak znakovnog niza (*string*). U funkciji je potrebno generirati znakovni niz duljine *n*, koji se sastoji od slučajno odabranih malih i velikih slova engleske abecede, a na koji će pokazivati *s*. Pretpostaviti da *s* pokazuje na memorijski prostor dovoljno velik za pohranu znakovnog niza duljine *n*.

c) Potrebno je napisati glavni program u kojem je potrebno učitati prirodne brojeve *m* i *n*, $m \leq 1000$ i $n \leq 100$. Brojeve *m* i *n* učitavati sve dok se ne učitaju ispravne vrijednosti.
Zatim je potrebno generirati *m* parova znakovnih nizova duljine *n* korištenjem funkcije `odrediSlucajniNiz`. Za svaki par nizova odrediti njihovu Hammingovu udaljenost korištenjem funkcije `HammingovaUdaljenost`.
Na kraju je potrebno ispisati tablicu s *n*+1 redaka (prema prikazanom predlošku) čiji su stupci redom: moguća Hammingova udaljenost (redom vrijednosti od 0 do *n*), broj parova generiranih nizova s tom vrijednosti udaljenosti, postoni udio takvih parova od ukupnog broja generiranih parova znakovnih nizova (*m*).
Primjer:
Za učitane vrijednosti *m* = 300 i *n* = 10, dobiven je rezultat ispisan prema sljedećem predlošku:

```
Hammingova udaljenost | broj pojavljivanja | postotak pojavljivanja
======================|====================|=========================
                    0 |                  0 |                    0.00
                    1 |                  0 |                    0.00
                    2 |                  0 |                    0.00
                    3 |                  0 |                    0.00
                    4 |                  0 |                    0.00
                    5 |                  0 |                    0.00
                    6 |                  0 |                    0.00
                    7 |                  0 |                    0.00
                    8 |                  6 |                    2.00
                    9 |                 41 |                   13.67
                   10 |                253 |                   84.33
```

19. zadatak

a) Napisati funkciju **`rasporedi`** koja će elemente iz zadanog jednodimenzionalnog polja pozitivnih cijelih brojeva nasumično rasporediti u dvodimenzionalno cjelobrojno polje pri čemu se elementi ne smiju stavljati preko već raspoređenih elemenata. Elementi dvodimenzionalnog polja na kojima nije raspoređen niti jedan element iz jednodimenzionalnog polja moraju imati vrijednost -1.

Funkcija preko ulaznih parametara prihvaća jednodimenzionalno polje pozitivnih cijelih brojeva, dvodimenzionalno cjelobrojno polje i ostale parametre potrebne za rad s jednodimenzionalnim i dvodimenzionalnim poljem.

Primjer: Neka je zadano jednodimenzionalno polje `2,3,4,7,8,9` koje je potrebno rasporediti u dvodimenzionalno polje s 4 retka i 3 stupca. Jedan mogući rezultat poziva funkcije može biti:

4	**9**	-1
-1	**2**	**8**
-1	-1	**3**
7	-1	-1

Napomena: Možete pretpostaviti da je broj elementa jednodimenzionalnog polja manji od maksimalnog broja elemenata dvodimenzionalnog polja zadanog brojem redaka i brojem stupaca.

b) Napisati funkciju **`generiraj`** koja će jednodimenzionalno polje napuniti nasumično generiranim pozitivnim cijelim brojevima iz zadanog intervala. Prototip funkcije je:

```
void generiraj(int* polje, int n, int dg, int gg);
```

c) Napisati glavni program u kojem je `s` tipkovnice potrebno učitati broj elemenata jednodimenzionalnog polja `n` te broj redaka `r` i broj stupaca `s` dvodimenzionalnog polja. Maksimalan broj elemenata jednodimenzionalnog polja je 100 a maksimalan broj redaka odnosno stupaca dvodimenzionalnog polja je 10. Zatim je potrebno nasumično generirati elemente jednodimenzionalnog polja iz intervala [20, 80] pomoću funkcije **`generiraj`** i pozvati funkciju **`rasporedi`** koja će elemente jednodimenzionalnog polja rasporediti u dvodimenzionalno polje. Na kraju je potrebno ispisati jednodimenzionalno polje i dvodimenzionalno polje.

20. zadatak

a) Napisati funkciju **generirajZnak** koja će nasumično generirati znak ovisno o zadanom parametru `vrsta`. Ovisno o vrijednosti parametra `vrsta` potrebno je generirati:

- Za vrijednost 1 generirati slovo engleske abecede
- Za vrijednost 2 generirati jednoznamenkasti broj
- Za vrijednost 3 generirati jedan od interpunkcijskih znakova **!**, **?** ili **/**

Prototip funkcije je:

```
char generirajZnak(int vrsta);
```

b) Napisati funkciju **generirajLozinku** koja će generirati lozinku koristeći funkciju **generirajZnak**. Prototip funkcije je

```
void generirajLozinku(int ns, int nb, int ni, char izl[]);
```

gdje je `ns` broj slova, `nb` broj brojeva i `ni` broj interpunkcijskih znakova koje lozinka mora sadržavati. Funkcija mora potreban broj slova, brojeva i interpunkcijskih znakova rasporediti nasumično u znakovnom nizu `izl`. Možete pretpostaviti da je veličina znakovnog niza dovoljno velika da u nju stane zadani broj slova, brojeva i interpunkcijskih znakova.

c) Napisati glavni program u kojem je s tipkovnice potrebno učitati broj slova, broj brojeva i broj interpunkcijskih znakova koje lozinka mora sadržavati. Pomoću funkcije **generirajLozinku** generirajte lozinku i ispišite ju na zaslon. Maksimalan broj znakova koje lozinka može sadržavati je 32.