Napomene:

1. Obavezno pročitati **SVE** napomene.
2. Zadatke snimiti pod imenom **`br_indeksa-1.S`**, **`br_indeksa-2.S`** i **`br_indeksa-3.S`**.

   Na primer: **`ra1234-2019-1.S`**, **`ra1234-2019-2.S`**, **`ra1234-2019-3.S`**

   Ovo su jedini fajlovi koji će biti pregledani! Rešenja NE smeju biti zapakovana u .zip ili neku drugu arhivu!
3. Obavezno upisati **ime**, **prezime** i **broj indeksa** u komentar na početku **SVAKOG** fajla.
4. Rešenje **mora** da se kompajlira kako bi ono bilo pregledano. Ukoliko se na primer zadaci 1 i 2 kompajliraju a zadatak 3 ne, studentu će prva dva zadatka biti pregledana.
5. Obavezno ostaviti komentare u kodu.
6. Za **zadatak 2**, u rešenju **NIJE DOZVOLJENO praviti sekcije podataka**. Ukoliko su potrebne dodatne promenljive, koristiti lokalne promenljive.
7. Nazivi promenljivih koje se inicijalno nalaze u zad.S fajlovima se **NE SMEJU menjati**. Sadržaji stringova sa porukama koje se inicijalno nalaze u u zad.S fajlu se **NE SMEJU menjati**. Vrednosti promenljivih se **mogu** (**i trebaju**) menjati, da se ispitaju razni ulazi za program.

Zadatak 1:

Napisati asemblerski program koji uneti string enkodira Cezarovim kodom. Podrazumevati da uneti string neće biti duži od 50 karaktera. Izlazni kod programa uvek treba da bude 0.

Cezarovo kodiranje u zadatku će rotirati znakove za tri mesta u levo i vrši se na sledeći način:

- Ako karakter nije slovo, ostaje u originalnom obliku, odnosno ne modifikuje se.
- Ako je karakter malo ili veliko slovo, njegova ASCII vrednost se smanji za 3 po modulu 26.

```
z (ASCII 122) -> w (ASCII 119 odnosno 122-3)
...
m (ASCII 109) -> j (ASCII 106 odnosno 109-3)
...
d (ASCII 100) -> a (ASCII 97 odnosno 100-3)
c (ASCII 99) -> z (ASCII 122 odnosno 99-3+26)
b (ASCII 98) -> y (ASCII 121 odnosno 98-3+26)
a (ASCII 97) -> x (ASCII 120 odnosno 97-3+26)
```

Primeri interakcije sa programom:

```
Unesite string: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Enkodovan string: XYZABCDEFGHIJKLMNOPQRSTUVW
```

```
Unesite string: Dobar dan zelim.
Enkodovan string: Alyxo axk wbifj.

Unesite string: Aly!xo a‘‘xk wbifj.
Enkodovan string: Xiv!ul x‘‘uh tyfcg.
```

Za kompletno urađen zadatak se dobija 20 poena.

Zadatak 2:

```
int encode(char* izvorni, char* ciljni, char* enkodovati);
```

Napisati potprogram koji enkodira string po korisnikovoj želji. Korisnik prvo unosi dva stringa (**izvorni_znakovi** i **ciljni_znakovi**) koji definišu enkripciju, i na kraju unosi string koji se enkodira (unos stringova se nalazi u datom C programu).
U stringu **izvorni_znakovi** korisnik unosi one znakove koje želi da transformiše u stringu koji enkodira.
U stringu **ciljni_znakovi** korisnik unosi znakove u koje želi da transformiše **izvorne znakove** u stringu koji se enkodira.
Podrazumevati da će stringovi **izvorni_znakovi** i **ciljni_znakovi** uvek imati isti broj znakova. U **izvornom** i **ciljnom** stringu se mogu naći i znakovi koji nisu slova. **Povratna vrednost potprograma** treba da bude broj znakova koji je zamenjen u stringu koji se enkodira.

izvorni_znakovi: a d o
↓ ↓ ↓
ciljni_znakovi: b e z

Nakon što korisnik unese string koji se enkodira, potrebno je proći kroz sve njegove znakove i ukoliko se neki od njih nalazi u stringu **izvorni_znakovi**, treba ga zameniti sa njegovim parom iz stringa **ciljni_znakovi**.
Na primer, ukoliko posmatramo gore navedene **izvorne** i **ciljne** znakove i u stringu za enkodovanje naiđemo na slovo ***d***, potrebno ga je zameniti sa njegovim parnjakom ***e***.

Primer interakcije sa programom:

```
Unesite izvorne znakove: Sie
Unesite ciljne znakove: bez
Unesite string za enkodovanje: String za izmenu.
String nakod enkodovanja:  btreng za ezmznu.

Unesite izvorne znakove: cdef
Unesite ciljne znakove: vghz
Unesite string za enkodovanje: abeceda i azbuka!
String nakod enkodovanja: abhvhga i azbuka!.
```

Za kompletno urađen zadatak se dobija 20 poena.

Zadatak 3:

Napisati asemblerski program koji predstavlja kalkulator logičkih operacija za 8-bitne brojve. Korisnik u program unosi jedan string maksimalne dužine 100 karaktera koji sadrži logički izraz oblika:

```
operand1 operacija1 operand2 operacija2 operand3 …
```

gde su operandi 8-bitne neoznačene vrednosti, a operacije jedan od binarnih logičkih operacija iz skupa [“^”,”<”,”>”] (isključivo ili, rotacija ulevo, rotacija udesno). Podrazumevati da će minimalno biti unesena dva operanda i jedna operacija između njih, dok se iza toga može nalaziti proizvoljan broj dodatnh operacija i opranada. Smatrati da će se između nizova znakova koji čine operande i operacije nalaziti tačno jedan razmak i da će broj takvih nizova znakova uvek biti neparan. Takođe, smatrati da se na početku i na kraju stringa neće unositi razmaci. Operandi se mogu zadati u dekadnom (npr. 123) ili heksadecimalnom (npr. 0x1f) obliku (“x” za heksadecimalni zapis može biti malo ili veliko slovo).

Računanje logičkog izraza ide s leva u desno, bez prioriteta operacija. Ukoliko prilikom računanja izraza nije bilo greške, rezultat treba ispisati u oktalnom brojnom sistemu. Ako se desi greška prilikom rada programa, prekinuti njegovo izvršavanje i ispisati poruku o grešci. Ako se desi greška kod prepoznavanja operanda, ispisati string **`err1`**. Ako se desi greska kod prepoznavanja operacije, ispisati string **`err2`**. Izlazni kod programa treba da bude **`0`** ako greške nije bilo, a **`1`** ako jeste.

Primeri korišćenja:

```
Unesite logicki izraz: 0xff < 36 ^ 0x6A > 11
Rezultat: 262

Unesite logicki izraz: 123 ^ 3f
Greska kod operanda.
```

Za kompletno urađen zadatak se dobija 30 poena.

--

Bodovanje zadataka će u velikoj meri zavisiti od procenta uspešnih testova.
Napomena: ukoliko testovi koji ne treba da izazovu grešku ne prolaze, testovi koji treba da izazovu grešku se ne uzimaju kao validni.

Pored testova koji su unapred dati (automatizovano testiranje sa **`./testiraj.sh zad.S`**), prilikom pregledanja rešenje će se testirati sa još dodatnih testova, te je potrebno testirati i sa drugim ulazima.
Napomena: **`testiraj.sh`** ima smisla pokretati tek kada je zadatak završen.