PRAKTIKUM IZ PROGRAMIRANJA 2

- domaći zadatak broj 5 -

Sastaviti program na programskom jeziku C kojim se vrši konverzija tekst datoteka sa podacima o kontaktima (eng. *address book*) iz jednog formata u drugi, uz određenu vrstu obrade nad podacima. Program treba da:

a) Pročita zadatu ulaznu datoteku;
b) Smesti podatke u odgovarajuću **jednostruko ulančanu** listu;
c) Izvrši odgovarajuću obradu nad ulaznim podacima;
d) Upiše rezultat u zadatu datoteku po tačno definisanom formatu u tekstu zadatka;
e) Dealocira svu alociranu dinamičku memoriju i zatvori korišćene datoteke.

Zavisno od rednog broja problema, svaki student treba da sastavi **jedan** od programa koji su dati u prilogu ovog dokumenta.

Opis formata ulaznih i izlaznih datoteka je dat u prilogu ovog dokumenta. Voditi računa o formatu izlazne datoteke definisanom u tekstu zadatka i priloženim primerima. Ukoliko korisnik zada praznu ulaznu datoteke, formirati praznu izlaznu datoteku.

Navedene korake u izvršavanju programa realizovati kao zasebne funkcije prema rasporedu navedenom u tekstu svakog od zadataka. Po potrebi, mogu se realizovati i druge, pomoćne funkcije koje vrše deo obrade. Student sam treba da definiše imena funkcija, kao i argumente potrebne za njihov rad. Voditi računa da se funkcijama dostave samo neophodni podaci (pokazivač na početak liste i, po potrebi, podatke od kojih zavisi obrada). **Potprogrami ne smeju komunicirati pomoću globalnih promenljivih, već samo preko liste argumenata i povratne vrednosti.** Glavni program treba da poziva funkcije koje obavljaju gorenavedene radnje. Sve funkcije smestiti u odgovarajuće **.c** datoteke (spisku u napomeni), a prototipove svih funkcija smestiti u zajedničku **.h** datoteku.

U zavisnosti od rednog broja problema koji se rešava osmisliti sopstvenu strukturu ili strukture podataka za smeštanje podataka iz ulazne datoteke. Element liste koja sadrži podatke iz ulazne datoteke treba realizovati kao **strukturu koja ima tačno dva polja – pokazivač na sledeći element i pokazivač na strukturu koja sadrži podatke**. Voditi računa o pravilnoj alokaciji i dealokaciji dinamičke memorije.

Program sa korisnikom treba da interaguje putem komandne linije. Korisnik kao prva dva, obavezna argumenta programa korisnik zadaje ime ulazne i izlazne datoteke. U zavisnosti od konkretnog zadatka, program može imati i dodatni opcioni argument.

U slučaju bilo kakve greške (poziv programa sa neodgovarajućim brojem ili vrstom argumenata komandne linije, neuspešna dodela dinamičke memorije, greška pri radu sa datotekom), ispisati poruku o grešci i korektno prekinuti izvršavanje (vraćanjem vrednosti 0 kao rezultata izvršavanja programa). U slučaju poziva programa sa neodgovarajućim brojem ili vrstom argumenata komandne linije, u programu ispisati poruku **`ARG_GRESKA`**. U slučaju neuspešne dodele dinamičke memorije, u programu ispisati poruku **`MEM_GRESKA`**. U slučaju greške pri radu sa datotekom, u programu ispisati poruku **`DAT_GRESKA`**. Zatim, korektno prekinuti izvršavanje programa.

Ako nešto u postavci zadatka nije dovoljno precizno definisano ili ako su neki od zahteva međusobno suprotstavljeni, usvojiti razumnu pretpostavku i rešiti zadatak korišćenjem te pretpostavke. Na samoj odbrani obavestiti demonstratora o usvojenoj pretpostavci ili pretpostavkama. Programski kod rešenja zadatka treba da bude uredno komentarisan, tako da pri pregledu programa lako može biti uočeno šta radi bilo koja programska celina. Takođe, treba poštovati pravila nazubljivanja (indentacije) određenih celina prilikom pisanja koda.

Radi boljeg testiranja programa, odabrati nekoliko skupova podataka sa kojima će program biti testiran. Svaki primer treba da sadrži ulazne podatke i očekivani izlaz za te podatke.

Napomene:

1. Rok za predaju petog domaćeg zadatka je **ponedeljak, 29.05.2023.** putem kursa predmeta na Moodle platformi za elektronsko učenje. Tačan termin za predaju će biti naknadno definisan za sve studente. Termin će biti ograničenog vremenskog trajanja.
2. Domaći zadaci će biti testirani i ocenjivani korišćenjem javnih i tajnih testova.
3. Studentima će nekoliko dana pre roka za predaju biti dostupno okruženje za testiranje rešenja domaćeg zadatka na Moodle platformi za elektronsko učenje korišćenjem javnih testova.
4. Prilikom predaje domaćeg zadatka studenti će rešavati i kratak test znanja u vezi rešenja domaćeg zadatka i relevantnog gradiva iz programskog jezika C koje obuhvata temu domaćeg zadatka. Na odbrani će biti postavljana pitanja, kao i dodatni zahtevi u vezi realizacije rešenja.
5. Domaći zadaci se rešavaju **samostalno**. Predmetni nastavnici zadržavaju pravo da nakon predaje domaćih zadataka izvrše proveru sličnosti i preduzmu odgovarajuće disciplinske mere.
6. Svi drugi detalji oko predaje i ocenjivanja domaćeg zadatka će biti blagovremeno objavljeni.
7. Formula za redni broj problema **i** koji treba rešavati je sledeća (R – redni broj indeksa, G – poslednje dve cifre godine upisa): **i = (R + G) mod 4**
8. Kao rešenje domaćeg zadatka potrebno je predati na Moodle kursu predmeta sadržaj sledeće datoteke:
    - **dz5.c**, koja sadrži izvorni tekst kompletnog programa na programskom jeziku C;
9. Kao rešenje domaćeg zadatka potrebno je na odbrani pokazati sledeće datoteke:
    - **dz5.h**, koja sadrži prototipe svih funkcija opisanih u postavci zadatka;
    - **dz5.c**, koja sadrži izvorni tekst osnovnog programa na programskom jeziku C;
    - **dz5_load.c, dz5_save.c, dz5_process.c**, koje sadrže izvorne tekstove funkcija potrebnih za inicijalizaciju programa, čitanje, snimanje i obradu kontakata;

22.05.2023. godine *sa predmeta*

0. Napisati program koji vrši konverziju podataka o kontaktima iz *Comma Separated Values* (.CSV) datoteke u *LDAP Data Interchange Format* (.LDIF) datoteku. Pored imena ulazne i izlazne datoteke, kao treći argument programa se opciono može navesti string na osnovu koga treba uraditi izostavljanje izlazne datoteke svih kontakata čiji se početak imena za prikaz (*Display Name*) poklapa sa zadatim stringom.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom kontaktu iz ulazne datoteke prema formatu koji je zadat u prilogu zadatka i formira strukturu podataka kojom se opisuje kontakt. Ukoliko ne pročita kontakt, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira ulančanu listu kontakata na način kako je to zahtevano tekstom zadatka.
   3) Implementirati funkciju koja uklanja iz liste sve kontakte čiji se početak imena za prikaz (*Display Name*) poklapa sa zadatim stringom.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke prema traženom formatu. Ime datoteke se zadaje kao jedan od argumenata funkcije.
   5) Implementirati funkciju koja oslobađa svu zauzetu dinamičku memoriju.
   6) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita ulaznu datoteku, izvrši zahtevanu obradu, formira izlaznu datoteku, dealocira svu korišćenu dinamičku memoriju i zatvori korišćene datoteke.

Primeri

Primer 1	
Poziv programa:	`program.exe ulaz.csv izlaz.ldif`
Ulazna datoteka:	`ulaz.csv`
Izlazna datoteka:	`izlaz.ldif`
Standardni izlaz:	
Primer 2	
Poziv programa:	`program.exe ulaz.csv izlaz2.ldif oo1pp2`
Ulazna datoteka:	`ulaz.csv`
Izlazna datoteka:	`izlaz2.ldif`
Standardni izlaz:	
Primer 3	
Poziv programa:	`program.exe ulaz.csv`
Ulazna datoteka:	`ulaz.csv`
Izlazna datoteka:	
Standardni izlaz:	`ARG_GRESKA`

1. Napisati program koji vrši konverziju podataka o kontaktima iz *Tab Separated Values* (.TXT) datoteke u *LDAP Data Interchange Format* (.LDIF) datoteku. Pored imena ulazne i izlazne datoteke, kao treći argument programa se opciono može navesti string na osnovu koga treba uraditi izostavljanje svih kontakata čije se prezime (*Last Name*) završava zadatim stringom.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom kontaktu iz ulazne datoteke prema formatu koji je zadat u prilogu zadatka i formira strukturu podataka kojom se opisuje kontakt. Ukoliko ne pročita kontakt, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira ulančanu listu kontakata na način kako je to zahtevano tekstom zadatka.
   3) Implementirati funkciju koja uklanja iz liste sve kontakte čije se prezime (*Last Name*) završava sa zadatim stringom.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke prema traženom formatu. Ime datoteke se zadaje kao jedan od argumenata funkcije.
   5) Implementirati funkciju koja oslobađa svu zauzetu dinamičku memoriju.
   6) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita ulaznu datoteku, izvrši zahtevanu obradu, formira izlaznu datoteku, dealocira svu korišćenu dinamičku memoriju i zatvori korišćene datoteke.

Primeri

Primer 1	
Poziv programa:	`program.exe ulaz.txt izlaz.ldif`
Ulazna datoteka:	`ulaz.txt`
Izlazna datoteka:	`izlaz.ldif`
Standardni izlaz:	
Primer 2	
Poziv programa:	`program.exe ulaz.txt izlaz2.ldif ic`
Ulazna datoteka:	`ulaz.txt`
Izlazna datoteka:	`izlaz2.ldif`
Standardni izlaz:	
Primer 3	
Poziv programa:	`program.exe ulaz.txt`
Ulazna datoteka:	`ulaz.txt`
Izlazna datoteka:	
Standardni izlaz:	`ARG_GRESKA`

2. Napisati program koji vrši konverziju podataka o kontaktima iz *Tab Separated Values* (.TXT) datoteke u *vCard Format* (.VCF) datoteku. Pored imena ulazne i izlazne datoteke, kao treći argument programa se opciono može navesti parametar **`-sort`** na osnovu koga treba uraditi uređivanje svih kontakata na osnovu polja imena za prikaz (*Display Name*) po neopadajućem leksikografskom poretku. Ukoliko dva kontakta imaju isto imena za prikaz (*Display Name*), zadržati isti relativan poredak između kontakata i u sortiranom poretku.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom kontaktu iz ulazne datoteke prema formatu koji je zadat u prilogu zadatka i formira strukturu podataka kojom se opisuje kontakt. Ukoliko ne pročita kontakt, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira ulančanu listu kontakata na način kako je to zahtevano tekstom zadatka.
   3) Implementirati funkciju koja sortira listu kontakata na osnovu polja imena za prikaz (*Display Name*) po neopadajućem leksikografskom poretku.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke prema traženom formatu. Ime datoteke se zadaje kao jedan od argumenata funkcije.
   5) Implementirati funkciju koja oslobađa svu zauzetu dinamičku memoriju.
   6) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita ulaznu datoteku, izvrši zahtevanu obradu, formira izlaznu datoteku, dealocira svu korišćenu dinamičku memoriju i zatvori korišćene datoteke.

Primeri

Primer 1	
Poziv programa:	`program.exe ulaz.txt izlaz.vcf`
Ulazna datoteka:	`ulaz.txt`
Izlazna datoteka:	`izlaz.vcf`
Standardni izlaz:	
Primer 2	
Poziv programa:	`program.exe ulaz.txt izlaz2.vcf -sort`
Ulazna datoteka:	`ulaz.txt`
Izlazna datoteka:	`izlaz2.vcf`
Standardni izlaz:	
Primer 3	
Poziv programa:	`program.exe ulaz.txt`
Ulazna datoteka:	`ulaz.txt`
Izlazna datoteka:	
Standardni izlaz:	`ARG_GRESKA`

3. Napisati program koji vrši konverziju podataka o kontaktima iz *Comma Separated Values* (.CSV) datoteke u *vCard Format* (.VCF) datoteku. Pored imena ulazne i izlazne datoteke, kao treći argument programa se opciono može navesti parametar **`-reverse`** na osnovu koga treba uraditi obrtanje redosleda svih kontakata u izlaznoj datoteci.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom kontaktu iz ulazne datoteke prema formatu koji je zadat u prilogu zadatka i formira strukturu podataka kojom se opisuje kontakt. Ukoliko ne pročita kontakt, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira ulančanu listu kontakata na način kako je to zahtevano tekstom zadatka.
   3) Implementirati funkciju koja obrće redosled elemenata prosleđene ulančane liste.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke prema traženom formatu. Ime datoteke se zadaje kao jedan od argumenata funkcije.
   5) Implementirati funkciju koja oslobađa svu zauzetu dinamičku memoriju.
   6) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita ulaznu datoteku, izvrši zahtevanu obradu, formira izlaznu datoteku, dealocira svu korišćenu dinamičku memoriju i zatvori korišćene datoteke.

Primeri

Primer 1	
Poziv programa:	`program.exe ulaz.csv izlaz.vcf`
Ulazna datoteka:	`ulaz.csv`
Izlazna datoteka:	`izlaz.vcf`
Standardni izlaz:	
Primer 2	
Poziv programa:	`program.exe ulaz.csv izlaz2.vcf -reverse`
Ulazna datoteka:	`ulaz.csv`
Izlazna datoteka:	`izlaz2.vcf`
Standardni izlaz:	
Primer 3	
Poziv programa:	`program.exe ulaz.csv`
Ulazna datoteka:	`ulaz.csv`
Izlazna datoteka:	
Standardni izlaz:	`ARG_GRESKA`