PRAKTIKUM IZ PROGRAMIRANJA 2
- domaći zadatak broj 5 -

Sastaviti program na programskom jeziku C kojim se vrši obrada tekst datoteka koje sadrže podatke o linijama gradskog prevoza u Beogradu. Program treba da:

a) Pročita jednu ili više zadatih ulaznih datoteka;
b) Smesti podatke u jednu ili više odgovarajućih **dvostruko ulančanih** lista;
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.

Podaci o jednoj liniji se nalaze u dve odvojene datoteke, za svaki smer posebno (**`dirA`** i **`dirB`**). Imena datoteka se sastoje od broja linije, donje crte (_) i oznake smera (**`dirA`** ili **`dirB`**). Podaci se u ulaznoj datoteci nalaze po sledećem formatu: broj stajališta (ceo broj), naziv stajališta (niz znakova), geografska širina (realan broj), geografska dužina (realan broj) i zona (ceo broj). Smatrati da znakovni nizovi nisu duži od 256 znakova. Primer nekoliko linija ulazne datoteke **`2_dirA.txt`** sledi. Polja su odvojena znakom uzvika (!).

Ulazna datoteka:	**2_dirA.txt**
7!Pristaniste!44.8189915!20.4495147!1 85!Masinski fakultet!44.8077358!20.476938!1 63!Vukov spomenik!44.8055219!20.4778908!1	

Podaci o geografskim koordinatama (geografskoj širini i dužini) stajališta su dati u sfernom koordinatnom sistemu. Ukoliko je potrebno, da bi se pronašlo rastojanje **`d`** između dve geografske koordinate, potrebno je primeniti **`haversine`** formulu:

$$t1 = sin^2\left(\frac{lat1 - lat2}{2}\right)$$

$$t2 = sin^2\left(\frac{lon1 - lon2}{2}\right)$$

$$d = 2 * R * \text{asin}\left(\sqrt{t1 + t2 * \cos(lat1) * \cos(lat2)}\right.$$

gde **`R`** predstavlja prečnik planete zemlje, **`lat1`** i **`lat2`** geografsku širinu prve, odnosno druge koordinate, a **`lon1`** i **`lon2`** geografsku dužinu prve, odnosno druge koordinate **u radijanima**. Smatrati da je prečnik planete Zemlje 6371 km. **Obratiti pažnju da su geografska širina (latitude) i geografska dužina (longitude) inicijalno zadate u stepenima, dok u `haversine` formuli treba da budu u radijanima.** Formula za pretvaranje stepeni u radijane je **`radians = degrees * 3.14 / 180.`**

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 i rezultata. Element liste koja sadrži podatke iz ulazne datoteke treba realizovati kao **strukturu koja ima tačno tri polja – pokazivače na prethodni i 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. Raspored i značenje pojedinačnih argumenata komandne linije su dati zasebno u okviru svakog zadatka.

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 (vraćanjem vrednosti 0 kao rezultata izvršavanja 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, 27.05.2024.** 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, ocenjivanja i odbrane 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 napraviti 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;

20.05.2024. godine *sa predmeta*

0. Napisati program koji pomaže putnicima u pronalaženju svih stajališta gradskog prevoza koja se nalaze u zadatom radijusu (poluprečniku) u odnosu na zadate koordinate putnika. Program putem komandne linije prihvata ime izlazne datoteke, smer (**`dirA`** ili **`dirB`**), koordinate putnika (geografsku širinu i dužinu), radijus (u metrima) i spisak linija gradskog prevoza koje su od interesa za putnika. Korisnik mora navesti oznaku bar jedne linije gradskog prevoza.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom stajalištu linije gradskog prevoza iz ulazne datoteke prema formatu koji je zadat u tekstu zadatka i formira strukturu podataka kojom se opisuje jedno stajalište linije gradskog prevoza. Ukoliko ne pročita stajalište, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira dvostruko ulančanu listu stajališta gradskog prevoza za zadatu liniju.
   3) Implementirati funkciju koja formira dvostruko ulančanu listu sa podacima o stajalištima zadatih linija koja se nalaze u zadatom radijusu (u metrima) od koordinata putnika. Ukoliko više linija deli isto stajalište, ono u listi treba da se pojavi samo jednom. Rastojanje između dve geografske tačke odrediti po datoj **`haversine`** formuli.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke. Ime datoteke se zadaje kao jedan od argumenata funkcije. Format jednog reda je isti kao kod ulazne datoteke, a datoteka treba da bude sortirana neopadajuće prema udaljenosti stajališta od putnika. Ukoliko su dva stajališta podjednako udaljena, sortirati ih leksikografski po nazivu.
   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 izlaz.txt dirA 44.8077350 20.476938 500 2 5 7 10`
Ulazna datoteka:	`2_dirA.txt 5_dirA.txt 7_dirA.txt 10_dirA.txt`
Izlazna datoteka:	`izlaz.txt`
Standardni izlaz:	-
Primer 2	
Poziv programa:	`program.exe izlaz.txt dirB 44.8055219 20.4778908 1000 12 27 32 25`
Ulazna datoteka:	`12_dirB.txt 27_dirB.txt 32_dirB.txt 25_dirB.txt`
Izlazna datoteka:	`izlaz.txt`
Standardni izlaz:	-
Primer 3	
Poziv programa:	`program.exe izlaz.txt`
Ulazna datoteka:	-
Izlazna datoteka:	-
Standardni izlaz:	`ARG_GRESKA`

1. Napisati program koji određuje rastojanja među stajalištima i ukupnu dužinu za zadate linije javnog gradskog prevoza i smer kretanja. Program putem komandne linije prihvata smer linije (**dirA** ili **dirB**) i spisak linija gradskog prevoza koje su od interesa za obradu. Korisnik mora navesti oznaku bar jedne linije gradskog prevoza. Program treba da izvrši zahtevanu obradu i formira zasebne izlazne datoteke sa podacima za svaku zadatu liniju. Ime izlazne datoteke formirati od imena linije, donje crte (_), smera, donje crte (_) i sufiksa **distance.txt**.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom stajalištu linije gradskog prevoza iz ulazne datoteke prema formatu koji je zadat u tekstu zadatka i formira strukturu podataka kojom se opisuje jedno stajalište linije gradskog prevoza. Ukoliko ne pročita stajalište, funkcija treba da vrati vrednost **NULL**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira dvostruko ulančanu listu stajališta gradskog prevoza za zadatu liniju.
   3) Implementirati funkciju koja utvrđuje pojedinačno rastojanje među susednim stajalištima i ukupnu dužinu linije za zadatu dvostruko ulančanu listu linije gradskog prevoza. Rastojanje između dve geografske tačke odrediti po datoj **haversine** formuli.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke za zadatu liniju gradskog prevoza. Ime datoteke se formira na opisani način za zadatu liniju. Format jednog reda je sledeći: **naziv stajališta 1! naziv stajališta 2!rastojanje**. Na kraj datoteke ispisati ukupnu dužinu linije u posebnom redu. Rastojanja zaokružiti na dve decimale. Stajališta i njihova rastojanja treba da budu u istom poretku kao što je ulazni.
   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 izlazne datoteke, dealocira svu korišćenu dinamičku memoriju i zatvori korišćene datoteke.

Primeri

Primer 1	
Poziv programa:	`program.exe dirA 2 5 7 10`
Ulazna datoteka:	`2_dirA.txt 5_dirA.txt 7_dirA.txt 10_dirA.txt`
Izlazna datoteka:	`2_dirA_distance.txt 5_dirA_distance.txt 7_dirA_distance.txt 10_dirA_distance.txt`
Standardni izlaz:	-
Primer 2	
Poziv programa:	`program.exe dirB 27 32`
Ulazna datoteka:	`27_dirB.txt 32_dirB.txt`
Izlazna datoteka:	`27_dirB_distance.txt 32_dirB_distance.txt`
Standardni izlaz:	-
Primer 3	
Poziv programa:	`program.exe dirB`
Ulazna datoteka:	-
Izlazna datoteka:	-
Standardni izlaz:	`ARG_GRESKA`

2. Napisati program koja za zadate linije javnog gradskog prevoza određuje stajališta sa istim nazivom u oba smera, kao i njihovu međusobnu udaljenost. Program putem komandne linije prihvata spisak linija gradskog prevoza koje su od interesa za obradu. Korisnik mora navesti oznaku bar jedne linije gradskog prevoza. Program treba da izvrši zahtevanu obradu i formira zasebne izlazne datoteke sa podacima za svaku zadatu liniju. Ime izlazne datoteke formirati od imena linije, donje crte (_) i sufiksa **`station.txt`**.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom stajalištu linije gradskog prevoza iz ulazne datoteke prema formatu koji je zadat u tekstu zadatka i formira strukturu podataka kojom se opisuje jedno stajalište linije gradskog prevoza. Ukoliko ne pročita stajalište, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira dvostruko ulančanu listu stajališta gradskog prevoza za zadatu liniju.
   3) Implementirati funkciju koja na osnovu dve dvostruko ulančane liste koje predstavljaju dva suprotna smera date linije formira listu svih onih stajališta zadate linije, čiji su nazivi isti i u jednom i drugom smeru, i određuje udaljenost između tih stajališta. Rastojanje između dve geografske tačke odrediti po datoj **`haversine`** formuli.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke za zadatu liniju gradskog prevoza. Ime datoteke se formira na opisani način za zadatu liniju. Format jednog reda je sledeći: **`naziv stajališta!udaljenost!zona`**. Stajališta treba da budu uređena po rastućem leksikografskom poretku svojih naziva. Udaljenost zaokružiti na dve decimale.
   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 izlazne datoteke, dealocira svu korišćenu dinamičku memoriju i zatvori korišćene datoteke.

Primeri

Primer 1	
Poziv programa:	`program.exe 2 5 7 10`
Ulazna datoteka:	`2_dirA.txt 5_dirA.txt 7_dirA.txt 10_dirA.txt` `2_dirB.txt 5_dirB.txt 7_dirB.txt 10_dirB.txt`
Izlazna datoteka:	`2_station.txt 5_station.txt 7_station.txt` `10_station.txt`
Standardni izlaz:	-
Primer 2	
Poziv programa:	`program.exe 27 32`
Ulazna datoteka:	`27_dirA.txt 32_dirA.txt 27_dirB.txt 32_dirB.txt`
Izlazna datoteka:	`27_station.txt 32_station.txt`
Standardni izlaz:	-
Primer 3	
Poziv programa:	`program.exe`
Ulazna datoteka:	-
Izlazna datoteka:	-
Standardni izlaz:	`ARG_GRESKA`

3. Napisati program koji pomaže putnicima u planiranju putovanja javnim gradskim prevozom od zadate početne lokacije do zadate krajnje lokacije korišćenjem zadatih linija javnog gradskog prevoza. Program putem komandne linije prihvata ime izlazne datoteke, koordinate početne i krajnje lokacije (geografsku širinu i dužinu) i spisak linija gradskog prevoza koje su od interesa za putnika. Korisnik mora navesti oznaku bar jedne linije gradskog prevoza. Program treba da izvrši zahtevanu obradu i formira izlaznu datoteke sa podacima za svaku zadatu liniju. Smatrati da putnik ne želi da menja liniju gradskog prevoza tokom putovanja.

   Program napisati prema sledećim stavkama.

   1) Implementirati funkciju koja čita podatke o jednom stajalištu linije gradskog prevoza iz ulazne datoteke prema formatu koji je zadat u tekstu zadatka i formira strukturu podataka kojom se opisuje jedno stajalište linije gradskog prevoza. Ukoliko ne pročita stajalište, funkcija treba da vrati vrednost **`NULL`**.
   2) Implementirati funkciju koja pozivanjem prethodno realizovane funkcije formira dvostruko ulančanu listu stajališta gradskog prevoza za zadatu liniju.
   3) Implementirati funkciju koja na osnovu dve prosleđene dvostruko ulančane liste formira listu stajališta zadate linije koje korisnik mora da pređe bi došao od zadate početne lokacije i stigao do zadate kranje lokacije. Kao početnu i krajnju stanicu uzeti one najbliže zadatim koordinatama. Prvo razmatrati samo stanice smera A, a nakon toga, ako putanja u smeru A ne postoji, razmatrati stanice smera B. Smatrati da korisnik ne može da prelazi iz jednog smera u drugi tokom svog putovanja. Rastojanje između dve geografske tačke odrediti po datoj **`haversine`** formuli.
   4) Implementirati funkciju koja vrši formiranje izlazne datoteke na osnovu formiranih lista stajališta za zadate linije. Ime datoteke se zadaje kao jedan od argumenata funkcije. Format jednog reda je isti kao kod ulazne datoteke. Pre ispisivanja putanje za određenu liniju ispisati u zasebnoj liniji oznaku linije i smer kretanja po formatu **`naziv linije!smer`**.
   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 izlaz.txt 44.777633 20.5299696 44.7569219 20.5519362 309`
Ulazna datoteka:	`309_dirA.txt 309_dirB.txt`
Izlazna datoteka:	`izlaz.txt`
Standardni izlaz:	-
Primer 2	
Poziv programa:	`program.exe izlaz.txt 44.7569219 20.5519362 44.795629 20.499284 309 7 46`
Ulazna datoteka:	`309_dirA.txt 309_dirB.txt 7_dirA.txt 7_dirB.txt 46_dirA.txt 46_dirB.txt`
Izlazna datoteka:	`izlaz.txt`
Standardni izlaz:	-
Primer 3	
Poziv programa:	`program.exe izlaz.txt`
Ulazna datoteka:	-
Izlazna datoteka:	-
Standardni izlaz:	`ARG_GRESKA`