PRAKTIKUM IZ PROGRAMIRANJA 2

- domaći zadatak broj 3 -

Sastaviti program na programskom jeziku C koji vrši određenu obradu nad jednim ili više **nizova znakova (u daljem tekstu stringova)**. Program treba da:

a) učita sve potrebne podatke;
b) izvrši zahtevanu obradu;
c) ispiše sve dobijene rezultate po tačno definisanom formatu u tekstu zadatka;

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

Prilikom ispisa na kraj svakog stringa dodati znak za novi red. Dimenzije pojedinačnih stringova nisu poznate unapred. Memorija predviđena za string mora biti **dinamički alocirana**. Količina memorije koja je odvojena za string mora biti takva da iskorišćenje na kraju učitavanja bude potpuno (ni bajt više, ni bajt manje). Realokaciju vršiti uvek kada je potrebno, i prilikom unosa stringa i prilikom obrade. U toku rada, prilikom svake alokacije ili realokacije dinamičke memorije, proveravati uspešnost poziva `alloc` funkcije. U slučaju neuspešne dodele dinamičke memorije, u glavnom programu ispisati poruku `MEM_GRESKA` i korektno prekinuti izvršavanje (vraćanjem vrednosti 0 kao rezultata izvršavanja programa).

Nakon obrade (unos, ispis unetih podataka, obrada, ispis dobijenih rezultata) dealocirati svu dinamičku memoriju. Smatrati da korisnik pojedinačan string zadaje u jednom redu teksta putem standardnog ulaza (u string ulaze svi znakovi do `\n`, ne uključujući i `\n`). Obradu treba vršiti nad originalnim stringom (ne nad kopijom), osim u slučaju formiranja novog stringa ili niza stringova, tamo gde je traženo.

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 drugog domaćeg zadatka je **četvrtak, 30.04.2020.** 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.
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 8**
8. Kao rešenje domaćeg zadatka potrebno je predati sadržaj sledeće datoteke:
   - **dz3.c**, koja sadrži izvorni tekst osnovnog programa na programskom jeziku C;

23.04.2020. godine *sa predmeta*

0. Napisati program koji formira novi string na osnovu zadatog šablona i zadatih stringova. Šablon se zadaje kao string u kojem je neophodno zameniti sve pojave znaka `'%'` konkretnim stringovima. Ukoliko šablon sadrži više znakova `'%'` nego unetih stringova, znakove `'%'` menjati stringovima ciklično. Ukoliko se ne unese nijedan string ispisati poruku **`GRESKA`**. Broj stringova, odnosno redova, nije unapred poznat. Kraj unosa se označava praznim redom.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readLine();`** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **`char** readLines(int *n);`** koja čita stringove kojima se menjaju sve pojave znaka `'%'` u šablonu. Funkcija kao povratnu vrednost vraća niz pokazivača na unete stringove. Takođe, funkcija preko argumenta **`n`** vraća broj unetih vrednosti.
   3) Implementirati funkciju **`char* format(char *format, char **values, int n);`** koja formatira string **`format`** tako što svaku pojavu znaka `'%'` menja odgovarajućim stringom iz niza stringova **`values`** dužine **`n`**.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita šablon i stringove sa standardnog ulaza, formira novi string na opisani način i ispiše rezultat na standardni izlaz. Nakon ispisa dodati znak za novi red.

Primeri

Ulaz:	**Izlaz:**

`% voli %`
`Ana`
`Milovana`
	`Ana voli Milovana`

`% ili ne %` `Biti`	`Biti ili ne Biti`
`% nema % vrednosti %`	`GRESKA`

1. Napisati program koji pronalazi sve pojave zadatih šablona u nekom stringu. String i šabloni se unose sa standardnog ulaza. U prvom redu se unosi string, dok se u narednim redovima unose šabloni za uparivanje. Broj redova nije unapred poznat. Kraj unosa se označava praznim redom. Šablon može sadržati specijalni znak `'%'` koji predstavlja džoker znak, odnosno odgovara tačno jednom alfanumeričkom karakteru ili blanko znaku. Prilikom uparivanja, ne uzimati u obzir preklapanja više pojava istog šablona. U slučaju da je string u kojem se traži šablon prazan ispisati poruku **`GRESKA`**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readLine();`** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **`char** readLines(int *n);`** koja čita šablone koji se traže u stringu. Funkcija kao povratnu vrednost vraća niz pokazivača na unete šablone. Takođe, funkcija preko argumenta **`n`** vraća broj unetih vrednosti.
   3) Implementirati funkciju **`char** findAll(char *string, char *pattern, int *n);`** koja pronalazi i vraća sve pojave šablona **`pattern`** u stringu **`string`**. Takođe, funkcija preko argumenta **`n`** vraća broj nađenih pojava.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita string i šablone sa standardnog ulaza, a zatim pronađe i ispiše sve pojave zadatih šablona u zadatom stringu. Nakon svake pojave staviti znak za novi red.

   Primeri

Ulaz:	**Izlaz:**
`Ana voli Milovana`	

`acb aca bcb`
`a%b`
`ca%b`
	`acb`
	`a b`
	`ca b`

`acb babb b`
`a%b%b`
`b%%b`
	`acb b`
	`abb b`
	`babb`

   Napomena

   U trećem primeru, sekvenca `bb b` nije uzeta u obzir, jer se delimično preklapa sa prethodnom pojavom zadatog šablona.

2. Napisati program koji u zadatom nizu stringova pronalazi i ispisuje sve parove stringova koji predstavljaju anagrame. Dva stringa se smatraju anagramima ukoliko se zamenom pozicija karaktera jednog stringa može dobiti drugi string. Broj stringova nije unapred poznat. Kraj unosa se označava praznim redom. Ukoliko je broj unetih stringova 0 ispisati poruku **`GRESKA`**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readLine();`** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **`char** readLines(int *n);`** koja vraća pokazivač na niz stringova pročitanih sa standardnog ulaza. Takođe, funkcija preko argumenta **`n`** vraća broj unetih stringova.
   3) Implementirati funkciju **`int areAnagrams(char *string0, char *string1);`** koja proverava da li stringovi **`string0`** i **`string1`** predstavljaju anagrame. Ukoliko predstavljaju funkcija vraća 1, a u suprotnom 0.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita niz stringova i ispiše sve parove stringova koji predstavljaju anagrame. Na kraju svakog reda dodati znak za novi red, a same stringove razdvojiti znakom **`'|'`**.

Primeri

Ulaz:	**Izlaz:**

`evil`
`vile`
`a gentleman`
`elegant man`
`pat`
`live`
	`evil|vile`
	`evil|live`
	`vile|live`
	`a gentleman|elegant man`

`one string`	
	`GRESKA`

3. Napisati program koji rotira rečenicu za zadati broj reči udesno ili ulevo. Na standardnom ulazu se najpre unosi rečenica u jednom ili više redova. Smatrati da su reči razdvojene tačno jednim blanko znakom. Kraj unosa se označava praznim redom. U sledećem redu se unosi broj reči za koji je neophodno rotirati rečenicu. Ukoliko je broj pozitivan rečenica se rotira udesno, a u suprotnom se rotira ulevo. Ukoliko se za rečenicu unese prazan string ispisati **GRESKA**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readWord();`** koja čita jednu reč sa standardnog ulaza i vraća pokazivač na tu učitanu reč. Čitanje se vrši do prvog blanko znaka ili do znaka za novi red.
   2) Implementirati funkciju **`char** readWords(int *n);`** koja vraća pokazivač na niz reči pročitanih sa standardnog ulaza. Takođe, funkcija preko argumenta **`n`** vraća broj unetih stringova.
   3) Implementirati funkciju **`void rotateSentence(char **sentence, int n, int rotate)`** koja rotira rečenicu za **`rotate`** reči. Rečenica je data nizom reči **`sentence`** dužine **`n`**.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita rečenicu sa standardnog ulaza, rotira je za zadati broj reči i ispiše je na standardnom izlazu. Reči razdvojiti blanko znakom bez znaka za novi red nakon poslednje reči.

Primeri

Ulaz:	**Izlaz:**
to be or not to be 3	not to be to be or
to be or not to be -2	or not to be to be
 2	GRESKA

4. Napisati program koji u zadatom nizu stringova pronalazi i menja sve datume u formatu **dd/mm/gg** u format **dd.mm.gggg.** tako da umesto **mm** stoji naziv meseca, a nakon datuma se dodaje jedan blanko znak i dan u nedelji na koji se datum odnosi. Smatrati da je **01.01.1970.** bio četvrtak, a da je godina prestupna ukoliko važi uslov (**gg%4 == 0 && gg % 100 != 0 || gg % 400 == 0**). Broj stringova nije unapred poznat. Smatrati da datum neće biti prelomljen u više redova. Kraj unosa se označava praznim redom. Ukoliko je broj unetih stringova 0 ispisati poruku **GRESKA**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **char* readLine();** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **char** readLines();** koja vraća pokazivač na niz stringova pročitanih sa standardnog ulaza. Kraj unosa se označava praznom linijom. Kraj niza pokazivača označava se **NULL** pokazivačem.
   3) Implementirati funkciju **int changeDateFormat(char **lines);** vrši izmene onih stringova u nizu **lines** koji sadrže datume u zadatom formatu. Povratna vrednost je broj takvih izmena.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita niz stringova, modifikuje one sa datumom u zadatom formatu i ispiše stringove u pojedinačnim redovima.

Primeri

Ulaz:	**Izlaz:**
Danas je 21/04/20, a sutra ce biti 22/04/20.	Danas je 21.April.2020. Utorak, a sutra ce biti 22.April.2020. Sreda.
Juce je bio 20/04/20, a za mesec dana ce biti 21/05/20.	Juce je bio 20.April.2020. Ponedeljak, a za mesec dana ce biti 21.Maj.2020. Cetvrtak.
	GRESKA

5. Napisati program koji u zadatom nizu stringova pronalazi i pomera za određeno vreme sva vremena u formatu **hh:mm**. Pomeraj je u formatu **hh:mm** sa znakom **+** (unapred) odnosno **–** (unazad) neposredno ispred pomeraja (npr. **+02:00**, **-3:30**). Vodeća nula ne mora postojati prilikom zadavanja vremena ili pomeraja, dok se kod ispisa mora navesti. Prvi red na standardnom ulazu označava pomeraj. Kraj unosa se označava praznim redom. Smatrati da vreme neće biti prelomljeno u više redova. Broj stringova nije unapred poznat. Kraj unosa se označava praznim redom. Ukoliko je broj unetih stringova 0 ispisati poruku **GRESKA**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readLine();`** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **`char** readLines();`** koja vraća pokazivač na niz stringova pročitanih sa standardnog ulaza. Kraj niza pokazivača označava se **NULL** pokazivačem.
   3) Implementirati funkciju **`void adjustTime(char **lines, char *time);`** vrši izmene onih stringova u nizu **lines** koji sadrže vreme u zadatom formatu **hh:mm** tako da se umesto sadržanog vremena nađe vreme pomereno za vreme **time**.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita niz stringova, modifikuje one sa vremenom u zadatom formatu i ispiše stringove u pojedinačnim redovima.

Primeri

Ulaz:	**Izlaz:**
+03:00 Treninzi ce se odrzati u 13:00, 15:00 i 22:00.	Treninzi ce se odrzati u 16:00, 18:00 i 01:00.
-2:00 Sastanak ce biti organizovan u 10:00 umesto u 12:00 kako je bilo predvidjeno.	Sastanak ce biti organizovan u 08:00 umesto u 10:00 kako je bilo predvidjeno.
	GRESKA

6. Napisati program koji u zadatom nizu stringova pronalazi i ispisuje sve parove stringova koji predstavljaju bliske stringove. Dva stringa su bliska ukoliko jedan može biti jednak drugom sa tačno jednom zamenom neka njegova dva karaktera. Broj stringova nije unapred poznat. Kraj unosa se označava praznim redom. Ukoliko je broj unetih stringova 0 ispisati poruku **GRESKA**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readLine();`** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **`char** readLines(int *n);`** koja vraća pokazivač na niz stringova pročitanih sa standardnog ulaza. Takođe, funkcija preko argumenta **`n`** vraća broj unetih stringova.
   3) Implementirati funkciju **`int areClose(char *string0, char *string1);`** koja proverava da li stringovi **`string0`** i **`string1`** bliski. Ukoliko predstavljaju funkcija vraća 1, a u suprotnom 0.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita niz stringova i ispiše sve parove stringova koji su bliski. Na kraju svakog reda dodati znak za novi red, a same stringove razdvojiti znakom `':'`.

Primeri

Ulaz:	**Izlaz:**

`live`
`meat`
`evil`
`team`
`beam`
`converse`
`meta`
`conserve`
	`meat:team`
	`meat:meta`
	`converse:conserve`

`give`
`vague`
`left`	

	`GRESKA`

7. Napisati program koji prelama tekst zadatog stringa, tako da nijedna linija teksta ne sadrži više od **len** karaktera (računajući i blanko znake između reči). Program najpre iz zadate linije teksta (stringa) vrši izdvajanje reči, a zatim formatira (prelama) tekst koristeći izdvojene reči. Sve reči u ulaznoj liniji odvojene su jednim blanko znakom. Nakon izdvajanja reči neophodno je izvršiti formatiranje teksta tako da svaka linija sadrži najviše **len** karaktera. Ukoliko je reč poslednja u liniji iza nje ne treba upisivati blanko znak. Dužina linije formatiranog teksta unosi se kao druga linija na standardnom ulazu. Prilikom formatiranja teksta po linijama, program treba da pokuša prelom reči po sledećem pravilu: prelom vršiti nakon odgovarajućeg samoglasnika, tako da uz dodavanje znaka – dužina linije bude najviše **len** karaktera. Svaku liniju treba što više popuniti. Ukoliko prelom nije moguć na opisani način, treba ga uraditi na mestu odgovarajućeg prethodnog blanko znaka. Ukoliko je neka od reči zadate linije teksta duža od dužine linije formatiranog teksta, a ne može se prelomiti ispisati poruku **GRESKA**.

   Program napisati prema sledećim stavkama. Prilikom izrade pojedinačne stavke pretpostaviti da su funkcije iz stavki koje su prethodile trenutnoj date i da njihove funkcionalnosti odgovaraju datom opisu.

   1) Implementirati funkciju **`char* readLine();`** koja čita jedan red sa standardnog ulaza i vraća pokazivač na taj učitani red.
   2) Implementirati funkciju **`char** splitLine(char *line, int *n);`** koja vraća pokazivač na niz stringova koji predstavljaju reči u liniji **line**. Takođe, funkcija preko argumenta **n** vraća broj reči.
   3) Implementirati funkciju **`char** formatText(char **words, int n, int len);`** koja vraća pokazivač na niz stringova koji predstavljaju linije formatiranog tekst. Dužina **len** predstavlja maksimalnu dužinu linije formatiranog teksta i unosi se kao druga linija na standardnom ulazu. Kraj niza pokazivača označava se **NULL** pokazivačem.
   4) Napisati glavni program koji korišćenjem prethodno realizovanih funkcija pročita jednu liniju teksta koja sadrži reči odvojene jednim blanko znakom, zatim izdvoji reči iz te linije i na osnovu njih kreira formatirani tekst, koji treba ispisati na standardnom izlazu.

Primeri

Ulaz:	**Izlaz:**

`This is some text that needs splitting`
`10`
	`This is`
	`some text`
	`that needs`
	`splitting`

`This is some text that needs spliting`
`12`
	`This is some`
	`text that`
	`needs spli-`
	`tting`

`Tvrd je orah vocka cudnovata` `3`	`GRESKA`