Pesak

Pažljivo pročitati sekciju Napomene.

Zadatak

Napisati program koji obrađuje podatke o skladištima peska. Podaci o skladištima i dimenzijama peska su zadati u ulaznoj datoteci u sledećem formatu:

- `sifra_skladista` (jedna reč, do 6 karaktera)
- `ukupna_sirina` (realan broj)
- `ukupna_duzina` (realan broj)
- `ukupna_visina` (celobrojna vrednost)

Učitati skladišta u jednostruko spregnutu listu, gde se novi čvorovi dodaju sortirani po zapremini peska u skladištu, u rastućem poretku.

Zapremina se računa na sledeći način:

$$zapremina = širina * dužina * visina$$

Program treba da sadrži implementacije sledećih funkcija (sa identičnim zaglavljima koja su ovde navedena, bez komentara koji opisuju šta funkcija radi):

```
// pokusava da otvori datoteku, izlazi iz programa u slucaju neuspešnog otvaranja
FILE *safe_fopen(char *naziv, char *rezim, int kod_greske);

// dinamicki alocira memoriju, uz odgovarajucu proveru,
// popunjava cvor sa vrednostima parametara funkcije i vraca adresu na novi cvor
SKLADISTE *napravi_cvor(char *sifra, float sirina, float duzina, float visina);

// ucitava podatke iz ulazne datoteke u jednostruko spregnutu listu, red po red
void ucitaj_skladista(FILE *ulazna, SKLADISTE **pglava);

// ispisuje na standardni izlaz sifre svih skladista cija visina je manja od prosledjene
void ispis_preniskih_skladista(SKLADISTE *glava, float trazena_visina);

// ispisuje najduzu kombinaciju skladista i njihovih zapremina koja se moze odjednom transportovati jednim kamionom unete zapremine,
// kao povratnu vrednost vraca preostalu zapreminu kamiona
// u slucaju da se nijedno skladiste ne moze transportovati jednim kamionom, ispisati na standardni izlaz "Sva skladista su veca od <ukupna_zapremina>!",
// u tom slucaju kao povratnu vrednost vratiti 0
float ispis_zapremina(FILE *pf, SKLADISTE *glava, float ukupna_zapremina);

// dodaje novi cvor u listu tako da lista bude sortirana po zapremini u rastucem redosledu
void dodaj_sortirano(SKLADISTE **pglava, SKLADISTE *novi);

// postavlja glavu liste na vrednost NULL
void inicijalizacija(SKLADISTE **pglava);

// dinamicki oslobadja memoriju svakog cvora (koristiti valgrind za proveru) i glavu liste postavlja na NULL
void obrisi_listu(SKLADISTE **pglava);
```

Primer ulazne i očekivane izlazne datoteke biće dati na samoj odbrani. Sastaviti testnu ulaznu datoteku i testirati rešenje pre predaje. Prilikom predaje, očistiti `main` funkciju, tako da izgleda na sledeći način:

```c
int main()
{
        return 0;
}
```

U slučaju uspešnog izvršavanja programa, izaći sa status kodom `0` (`EXIT_SUCCESS`). Ukoliko program ne može da se izvrši do kraja usled sledećih nedostataka, izaći iz programa sa sledećim status kodovima:

- U slučaju nedovoljnog ili suvišnog broj argumenata komandne linije, izaći iz programa sa status kodom `1` (`EXIT_FAILURE`)

- U slučaju nemogućnosti dinamičkog zauzimanja memorije, izaći iz programa sa status kodom `2`
- Ako program ne može da otvori ulazni fajl, izaći sa status kodom `3`
- Ako program ne može da otvori izlazni fajl, izaći sa status kodom `4`
- Ako neko od skladišta ima zapreminu manju ili jednaku `0`, izaći sa status kodom `5`

Napomene

- Programski kod ne sme sadržati sintaksne greške, niti upozorenja
- Programski kod mora biti snimljen u okviru home direktorijuma, na predviđenom mestu
- Nije dozvoljeno snimanje više kopija ili varijacija izvornog koda, samo jedna .c datoteka
- Prilikom rada programa poštovati očekivani format teksta, prikazanom u tekstu zadatka
- Nije dozvoljeno koristiti globalne promenljive
- Obavezno osloboditi dinamički zauzetu memoriju
- Funkcije treba obavezno da poštuju zaglavlja zadata u zadatku
- Testirati sve detaljno pre predaje, ali rešenje predati sa praznom main funkcijom (bez parametara za argumente komandne linije, samo sa return 0 naredbom)
- Podrazumevati da su sve merne jedinice iste