**ELEKTROTEHNIČKI FAKULTET**
**BANJA LUKA**

# PROGRAMIRANJE II (16.06.2020.)

❶ (**15 bodova**) Napisati rekurzivnu funkciju koja kao rezultat vraća ukupan broj slova (velikih i malih) u stringu. Svi ostali znakovi, izuzev slova, se ignorišu. Svaki znak stringa se procesira (obrađuje) u odvojenom rekurzivnom pozivu. Prototip funkcije je:

**int alpha_num(const char *str);**

Napisati rekurzivnu funkciju koja sabira sve cifre u stringu (svaka cifra je poseban broj tj. brojevi se posmatraju kao jednocifreni). Svaki znak stringa se procesira (obrađuje) u odvojenom rekurzivnom pozivu. Prototip funkcije je:

**int number_sum(const char *str);**

Napisati funkciju koja kao neobavezne argumente prihvata stringove te nad istim poziva funkciju koja je prosleđena kao argument. Pri tome funkcija sumira rezultate poziva funkcije koja je prosleđena kao argument, te sumu vraća kao rezultat. Prototip funkcije je:

**int total_sum(int (*t)(const char *), int n, ...);**

U glavnom programu ilustrovati upotrebu funkcije sa neobaveznim argumentima nad minimalno tri stringa za oba slučaja prethodno navedenih rekurzivnih funkcija.

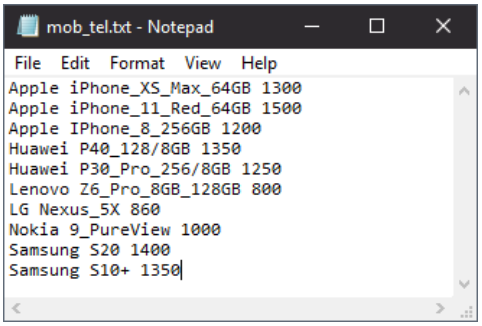
❷ (**15 bodova**) U tekstualnoj datoteci, čiji se naziv prosleđuje kao prvi argument komandne linije, smešteni su podaci o nepoznatom broju mobilnih telefona. Telefoni su leksikografski sortirani po proizvođaču telefona.

Potrebno je napisati program u kojem se:

- učitavaju podaci iz tekstualne datoteke i kreira dinamički niz,
- korištenjem binarnog pretraživanja pronalazi indeks elementa u nizu čiji proizvođač odgovara zadatoj tekstualnoj vrednosti, koja se unosi putem standardnog ulaza,
- na standardni izlaz ispisuje pronađeni mobilni telefon, kao i svi mobilni telefoni istog proizvođača koristeći prethodno pronađeni indeks.

Za svaki telefon se čuvaju sledeći podaci: proizvođač, model i cena.

Primer datoteke:




❸ (**15 bodova**) Neka je dat tip:

```
typedef struct node {
   char str[64];
   struct node *next;
} NODE;
```

kojim se reprezentuje čvor jednostruko ulančane liste stringova.

Napisati funkciju koja dodaje novi string na početak (neprazne) liste. Funkcija vraća adresu novog čvora ili NULL ako novi čvor nije dodan. Ignorisati pokušaj dodavanja stringa u sledećim slučajevima: (1) ako je lista prazna, (2) ako parametar str ima vrednost NULL, (3) ako je string prazan, (4) ako je string predugačak. Prototip funkcije je:

**NODE* str_add(NODE *head, const char *str);**

Napisati funkciju koja vraća ukupnu dužinu svih stringova u listi, a čiji je prototip:

**int str_length(NODE *head);**

Napisati funkciju koja vraća dinamičku kopiju najdužeg stringa u listi. Ako lista sadrži više stringova iste (najveće) dužine, funkcija treba da vrati kopiju prvog takvog stringa. Ako je lista prazna, funkcija treba da vrati vrednost NULL. Prototip funkcije je:

**char* str_max_length(NODE *head);**

❹ (**15 bodova**) Neka su definisani tipovi:

```
typedef struct {
   char ime[64];
} OSOBA;
```

i

```
typedef struct {
   int n; // broj osoba/cvorova
   OSOBA cvor[100]; // inf. sadrzaj
   int ms[100][100];
} GRAF;
```

kojima se reprezentuje (mala) društvena mreža (usmeren graf) sa najviše sto osoba. Pri tome, jedan čvor društvene mreže reprezentuje jednu osobu, a usmerena grana od čvora (osobe) A do čvora (osobe) B znači da osoba A prati osobu B na datoj društvenoj mreži. Takođe, svaka osoba na mreži ima jedinstveno ime.

Napisati funkciju koja vraća indeks čvora grafa g, koji reprezentuje osobu čije je ime parametar funkcije. U slučaju da osoba sa datim imenom nije na mreži, funkcija treba da vrati vrednost -1. Prototip funkcije je:

**int pozicija(GRAF *g, const char *ime);**

Napisati funkciju čiji su parametri ime osoba A (parametar ime_a) i ime osobe B (parametar ime_b), a koja treba da evidentira činjenicu da je osoba A zapratila osobu B. Funkcija treba da vrati vrednost 1 ako je evidentiranje uspešno. U suprotnom, ako osobe A i B nisu na mreži ili osoba A već prati osobu B, funkcija treba da vrati vrednost 0. Prototip funkcije je:

**int zaprati(GRAF *g, const char *ime_a, const char *ime_b);**

Napisati funkciju koja vraća ukupan broj posećenih čvorova grafa prilikom obilaska po dubini (DFS), pri čemu obilazak treba da započene od čvora koji reprezentuje osobu čije je ime parametar funkcije. Ako data osoba nije na mreži, funkcija treba da vrati vrednost -1. Implementacija obilaska treba da bude rekurzivna, pri čemu je dozvoljeno definisanje dodatnih funkcija. Prototip funkcije je:

**int mreza(GRAF *g, const char *ime);**