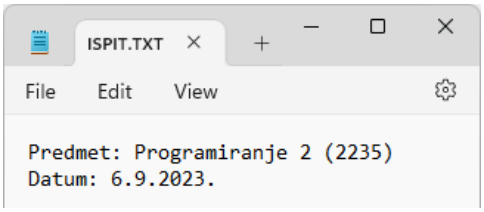
ELEKTROTEHNIČKI FAKULTET
BANJA LUKA

PROGRAMIRANJE II
(06.09.2023)

❶ **(20 bodova**) Napisati funkciju koja iz ulazne tekstualne datoteke (dat) učitava i vraća sve prirode brojeve. Prirodan broj (u datoteci) je jedna ili više uzastopnih cifara, a od drugog prirodnog broja odvojen je jednim ili više znakova koji nisu cifre. Pretpostaviti da se svaki prirodan broj može reprezetovati tipom int. Preko parametra *n* funkcija vraća ukupan broj učitanih prirodnih brojeva. Prototip funkcije je:

int* prirodni(FILE *dat, int *n);

Na primjer, u tekstualnoj datoteci na slici upisani su prirodni brojevi: 2, 2235, 6, 9 i 2023.



Napisati rekurzivnu funkciju koja sabira niz od *n* prirodnih brojeva, a čiji je prototip:

int zbir(const int *niz, int n);

Napisati funkciju sa promjenljivim brojem argumenata koja prihvata prirodan broj *k* i *k* stringova koji predstavljaju nazive ulaznih tekstualnih datoteka. Funkcija treba da izračuna i vrati zbir svih prirodnih brojeva koji su upisani u ulaznim tekstualnim datotekama. Prototip funkcije je:

int zbir_d(int k, ...);

❷ **(20 bodova**) Napisati funkciju koja sortira u rastućem redoslijedu (po nekom kriterijumu) niz od *n* stringova. Sortiranje treba izvršiti *insert-sort* algoritmom. Prototip funkcije je:

void sortiraj(char **niz, int n, int (*cmp)(const char *, const char *));

Napisati nerekurzivnu funkciju koja binarno pretražuje (po nekom kriterijumu) niz od *n* stringova, pri čemu je ključ pretrage zadat kao parametar funkcije (*kljuc*). U slučaju uspješne pretrage funkcija vraća indeks pronađenog elementa, a u slučaju neuspješne pretrage funkcija vraća -1. Prototip funkcije je:

int pretrazi(char **niz, int n, int (*cmp)(const char *, const char *), const char *kljuc);

U glavnom programu treba formirati niz od pet proizvoljnih stringova, a zatim ga sortirati (korištenjem funkcije *sortiraj*). Na kraju, provjeriti (korištenjem funkcije *pretrazi*) i ispisati da li se u sortiranom nizu nalazi proizvoljan string. Kao kriterijum za poređenje koristiti dužinu stringa.

Napomena: U funkcijama *sortiraj* i *pretrazi*, parametar *cmp* predstavlja pokazivač na funkciju koja vrši poređenje dva stringa prema nekom kriterijumu. Funkcija za poređenje vraća negativnu vrijednost ako je prvi string manji od drugog, vrijednost 0 ako su stringovi jednaki te pozitivnu vrijednost ako je prvi string veći od drugog.

❸ **(20 bodova)** Neka je dat tip:

```
typedef struct cvor {
    char str[64];
    struct cvor *sljedeci;
} CVOR;
```

kojim se reprezentuje čvor ulančane reprezentacije steka.

Napisati funkciju koja dodaje novi string na stek. Funkcija vraća adresu novog čvora ili NULL ako novi čvor nije dodan.

Ignorisati pokušaj dodavanja stringa u sljedećim slučajevima: (1) ako je stek prazan, (2) ako parametar *str* ima vrijednost NULL, (3) ako je string prazan, (4) ako je string predugačak. Prototip funkcije je:

CVOR* str_add(CVOR **tos, const char *str);

Napisati funkciju koja vraća ukupnu dužinu svih stringova na steku, a čiji je prototip:

int str_length(CVOR **tos);

Napisati funkciju koja vraća dinamičku kopiju najdužeg stringa na steku. Ako stek sadrži više stringova iste (najveće) dužine, funkcija treba da vrati kopiju prvog takvog stringa.

Ako je stek prazan, funkcija treba da vrati vrijednost NULL. Prototip funkcije je:

char* str_max_length(CVOR **tos);

Napomena: Nakon poziva funkcija *str_length* i *str_max_length* stek mora biti prazan.

❹ **(20 bodova)** Neka je definisan tip:

```
typedef struct node {
    int info;
    struct node *next;
} NODE;
```

kojim se reprezentuje čvor jednostruko ulančane liste.

Neka je definisan tip:

```
typedef struct graph {
    int n; // broj cvorova
    NODE **al; //liste susjednosti
} GRAPH;
```

koji predstavlja ulančanu reprezentaciju usmjerenog grafa kod kojeg liste susjednosti ne moraju biti uređene.

Napisati funkciju koja za predefinisani čvor računa broj čvorova u susjedstvu koji se nalaze na *n* skokova ili manje. Na primjer, ako računamo za *n=3*, to bi obuhvatalo sve čvorove koji se nalaze na udaljenosti od 1, 2 i 3 skoka. Prototip funkcije je:

int broj_n_hop_susjeda(GRAPH *g, int cvor, int n);

Napomena: Dozvoljeno je koristiti i *DFS* I *BFS* implementaciju za dati problem. U oba slučaja je potrebno definisati pomoćnu funkciju koja bi vršila računanje.