ELEKTROTEHNIČKI FAKULTET
BANJA LUKA

PROGRAMIRANJE II (24.04.2018.)

❶ U ulaznu binarnu datoteku upisani su neki podaci. Napisati funkciju koja kompresuje ulaznu binarnu datoteku (*in*), bez učitavanja sadržaja datoteke u memoriju, tako što svaki niz od jednog ili više istih uzastopnih bajtova, mijenja sa dva podatka: broj uzastopnih ponavljanja (jednobajtni podatak) datog bajta, te taj bajt. Rezultat kompresije treba da se upiše u drugu binarnu datoteku (*out*), tj. funkcija ne modifikuje ulaznu binarnu datoteku. Na primjer, ako je sadržaj ulazne binarne datoteke:

```
61 61 61 61 49 49 49 31 61 61 61 61 61
```

tada bi sadržaj izlazne datoteke bio:

```
04 61 03 49 01 31 05 61
```

Prototip funkcije je:

```
void compress(FILE *in, FILE *out);
```

Napisati funkciju koja kao obavezni parametar prihvata broj *n*, a zatim *n* stringova (neobavezni parametri) koji reprezentuju nazive ulaznih binarnih datoteka. Funkcija treba da kompresuje ulazne binarne datoteke (korištenjem funkcije *compress*), pri čemu se naziv izlazne datoteke dobija dodavanjem prefiksa "compressed_" na naziv ulazne datoteke. Prototip funkcije je:

```
void compress_n(int n, ...);
```

U glavnom programu ilustrovati korištenje funkcije *compress_n* tako što treba izvršiti kompresiju tri binarne datoteke čiji su nazivi argumenti komandne linije.

❷ Napisati funkciju koja *insert-sort* algoritmom vrši indirektno sortiranje polazne kolekcije podataka (funkcija ne vrši promjenu rasporeda u datoj polaznoj kolekciji, nego se formira i vraća (dinamička) uređena kolekcija indeksa). Prototip funkcije je:

```
int* isort(const void *a, int n, int size, int
(*cmp)(const void *, const void *));
```

pri čemu je: *a* - polazna kolekcija, *n* - broj elemenata polazne kolekcije, *size* - veličina jednog podatka polazne kolekcije, *cmp* - funkcija za poređenje dva elementa polazne kolekcije.

U glavnom programu definisati i inicijalizovati niz od deset cijelih brojeva. Korištenjem funkcije *isort* sortirati, a zatim ispisati sortirani niz cijelih brojeva.

❸ Napisati rekurzivnu funkciju čiji je prototip:

```
void prekopiraj(FILE *f1, char *niz, int n, int
*kd);
```

koja iz otvorene tekstualne datoteke *f1* preuzima po *n* karaktera i smiješta tih *n* karaktera na kraj niza, tj. promjenljive *niz*. U promjenljivu *kd* (početna vrijednost na koju pokazuje je 0) potrebno je smjestiti konačan broj elemenata niza. Dozvoljen je samo jedan pristup datoteci u jednom rekurzivnom pozivu. Smatrati da niz prije poziva funkcije nije alociran.

❹ Napisati program kojim se porede brzine izvršavanja algoritama za sekvencijalnu pretragu sa stražom i binarnu pretragu, na istom nizu pseudoslučajnih vrijednosti. Niz pseudoslučajnih vrijednosti potrebno je, prije pretrage, sortirati korištenjem *qsort* funkcije (funkcija iz *stdlib.h*). Kao ključ pretrage, potrebno je uzeti prvu učitanu pseudoslučajnu vrijednost.

❺ Neka su date sljedeće strukture:

```
typedef struct stek {
  CVOR *liste[MAX];
  int tos;
} STEK;

typedef struct cvor {
  char *rijec;
  struct cvor *sljedeci;
} CVOR;
```

koje predstavljaju stek i čvor jednostruko povezane liste, respektivno, pri čemu sadržaj steka čine liste stringova koji počinju istim slovom.

Napisati funkciju koja dodaje novi string *s* u odgovarajuću listu na steku, pri čemu je string potrebno dodati na kraj liste. Ukoliko ne postoji lista stringova koji počinju istim slovom kao string *s*, onda je potrebno kreirati novu listu na steku i u nju dodati string *s*. Prototip funkcije je:

```
void push_s(STEK *stek, char *s);
```

Napisati funkciju koja skida sa steka listu stringova koji počinju slovom *sl*. Ukoliko takva lista ne postoji na steku, potrebno je ispisati odgovarajuću poruku na standardni izlaz. Prototip funkcije je:

```
int pop_l(STEK *stek, char sl, CVOR **glava);
```

Sve dodatne pomoćne funkcije, u slučaju korištenja, potrebno je definisati.

❻ Neka je data matrična reprezentacija težinskog grafa. Definisati datu strukturu *GRAF* te napisati funkciju čiji je prototip:

```
int izvagaj(GRAF *g, int a, int b);
```

Funkcija za dati graf g i indekse čvorova u grafu (vrijednosti *a* i *b*) traži putanju između ta dva čvora koja ima najveću težinsku vrijednost te, kao rezultat, vraća ukupnu težinu te putanje. Ukoliko ne postoji putanja, funkcija vraća -1.

Maksimalan broj bodova po zadacima

Integralno				
1.	2.	5.	6.	Σ
25	25	25	25	**100**

K1				
1.	2.	3.	4.	Σ
25	25	25	25	**100**