ELEKTROTEHNIČKI FAKULTET
BANJA LUKA

PROGRAMIRANJE II (10.07.2018.)

❶ Neka su definisani tipovi:

```
typedef struct zaposleni {
  char jmb[14], ime[21], prezime[21];
  double plata;
} ZAPOSLENI;

typedef struct indeks {
  char kljuc[14];
  int adresa;
} INDEKS;
```

pri čemu *INDEKS* predstavlja *look-up* tabelu za efikasniju pretragu podataka o zaposlenima. Polje *kljuc* predstavlja JMBG zaposlenog, a *adresa* predstavlja adresu početka zapisa o odgovarajućem zaposlenom u datoteci koja sadrži podatke o zaposlenima. U datoteci sa indeksima nalaze se indeksni zapisi (smatrati da su zapisi sortirani prema vrijednosti polja *kljuc*), pri čemu se svaki zapis nalazi u novom redu, a jedan zapis je oblika:

```
kljuc#adresa
```

Napisati funkciju koja iz datoteke sa indeksima čita sve indeksne zapise i formira dinamički niz, a čiji je prototip:

```
int readI(FILE *f, INDEKS **niz);
```

Funkcija vraća ukupan broj pročitanih zapisa.

Napisati funkciju za binarnu pretragu indeksa koja ima sljedeći prototip:

```
int bSearch(INDEKS *niz, int n, char *kljuc,
int (*cmp)(const void *, const void *));
```

pri čemu je *cmp* pokazivač na funkciju koja poredi dva podatka nekog tipa, a koju je potrebno iskoristiti u funkciji *bSearch*.

U glavnom programu demonstrirati pretragu podataka o zaposlenima, pri čemu se JMBG zaposlenog navodi kao prvi, a nazivi tekstualnih datoteka sa zaposlenima i indeksima kao drugi i treći argument komandne linije. Pretraživanje je neophodno realizovati bez učitavanja sadržaja datoteke sa zaposlenima u memoriju.

❷ Napisati funkciju koja ima sljedeći prototip:

```
void napravi_niz(char ***niz, int n, ...);
```

Funkcija kao argumente prima niz stringova *niz* (funkcija treba da alocira prostor za novokreirani niz stringova), broj stringova *n*, te stringove kao neobavezne argumente, respektivno.

U sklopu funkcije potrebno je stringove koji su neobavezni argumenti smjestiti sortirane u opadajućem redoslijedu alfanumerički u niz stringova tako da je svaki string jedan element novokreiranog niza. Za sortiranje koristiti *insertion-sort* algoritam.

U glavnom programu demonstrirati upotrebu prethodno kreirane funkcije te nakon toga ispisati novokreirani niz stringova na standardni izlaz (ispis je u glavnom programu).

❸ Neka je definisan tip:

```
typedef struct node {
  void *data; // dinamicki podatak
  struct node *next;
} NODE;
```

kojim se reprezentuje čvor jednostruko ulančane liste podataka proizvoljnog tipa, te tip:

```
typedef struct sll {
  NODE *head, *tail;
} SLL;
```

kojim se reprezentuje jednostuko ulančana lista podataka proizvoljnog tipa.

Napisati funkciju koja dodaje novi podatak (*data*) na kraj liste, a čiji je prototip:

```
void add(SLL *list, void *data);
```

Napisati funkciju koja provjerava da li su dvije liste podataka istog tipa jednake (dvije liste su jednake ako imaju isti broj elemenata i ako su korespondentni parovi elemenata jednaki). Prototip funkcije je:

```
int equals(const SLL *l1, const SLL *l2, int
(*cmp)(const void *, const void *));
```

pri čemu je *cmp* pokazivač na funkciju koja poredi dva podatka nekog tipa, te vraća vrijednost <0 ako je prvi podatak manji od drugog, 0 ako su podaci jednaki, a >0 ako je prvi podatak veći od drugog. Funkcija *equals* treba da vrati vrijednost 1 ako su liste jednake; u suprotnom, funkcija treba da vrati vrijednost 0.

U glavnom programu, korištenjem prethodno definisanih funkcija, potrebno je formirati dvije liste cijelih brojeva koji se učitavaju sa standardnog ulaza, a zatim na standardni izlaz ispisati da li su učitane liste jednake.

❹ Neka je definisan tip:

```
typedef struct cvor {
  int i;
  struct cvor *s;
} CVOR;
```

kojim se reprezentuje čvor jednostruko ulančane liste cijelih brojeva, te tip:

```
typedef struct graf {
  int n;       // broj cvorova
  char *r[10]; // inf. sadrzaj cvorova
  CVOR *s[10]; // liste susjednosti
} GRAF;
```

kojim se reprezentuje neusmjeren graf (ulančana reprezentacija).

Neka je definisana funkcija koja dodaje novi element u jednostruko ulančanu listu, a čiji je prototip:

```
void dodaj(CVOR **pglava, int i);
```

Napisati funkciju koja formira i vraća obuhvatno stablo obilaskom grafa *g* (pretpostaviti da je graf povezan) po dubini, počevši od čvora *c*. Prototip funkcije je:

```
GRAF* st(const GRAF *g, int c);
```

U glavnom programu je potrebno formirati povezan graf sa pet čvorova, a zatim formirati obuhvatno stablo počevši od prvog čvora (čvor sa indeksom 0). Na standardni izlaz ispisati liste susjednosti formiranog obuhvatnog stabla.

Maksimalan broj bodova po zadacima

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

K1			**K2**		
1.	2.	**Σ**	3.	4.	**Σ**
25	25	**50**	25	25	**50**