ELEKTROTEHNIČKI FAKULTET
BANJA LUKA

PROGRAMIRANJE II (23.09.2016.)

❶ (**25 bodova**) Neka je definisan tip:

```
typedef struct krug
{
  double x, y, r;
} KRUG;
```

kojim se reprezentuje krug, čiji su atributi *x* i *y* koordinate centra kruga, a *r* poluprečnik kruga.

Definisati funkciju koja sortira niz krugova, rastuće, po obimu. Za sortiranje koristiti *shell-sort* algoritam. Prototip funkcije je:

```
void shell_sort(KRUG *, int);
```

Definisati funkciju koja iz ulazne tekstualne datoteke, čiji je naziv prvi argument funkcije, čita podatke o nepoznatom broju krugova. Funkcija formira i vraća dinamički niz sortiranih (koristiti funkciju *shell_sort*) krugova. Prototip funkcije je:

```
KRUG* formiraj(char *, int *);
```

Napomena: U svakom redu ulazne datoteke upisan je podatak o jednom krugu u obliku:

```
(x,y,r)
```

❷ (**30 bodova**) Potrebno je implementirati jednostruko povezanu listu koja sadrži podatke o krugovima. Neka je definisan tip:

```
typedef struct cvor
{
  KRUG k;
  struct cvor *sljedeci;
} CVOR;
```

kojim se reprezentuje čvor jednostruko povezane liste. Čvor liste sadrži podatke o krugu, pri čemu je potrebno koristiti definiciju tipa KRUG iz prethodnog zadatka.

Definisati funkciju koja dodaje novi čvor na kraj liste, pri čemu elementi liste moraju biti jedinstveni. Prototip funkcije je:

```
void dodaj(CVOR **pglava, KRUG k);
```

Definisati funkciju koja formira i vraća novu listu koja predstavlja presjek (sadrži samo krugove koji se nalaze u obe liste, pri čemu su dva kruga jednaka ako imaju isti poluprečnik) dvije liste. Vremenska složenost formiranja presjeka treba biti linearna. Prototip funkcije je:

```
CVOR* presjek(CVOR *glava1, CVOR *glava2);
```

U glavnom programu formirati dvije liste koje se sastoje od krugova pročitanih iz tekstualnih datoteka čiji su nazivi prvi i drugi argument komandne linije (koristiti funkciju *formiraj* iz prethodnog zadatka). Za formiranje listi koristiti funkciju *dodaj*. Korištenjem funkcije *presjek* pronaći presjek dvije liste, i elemente rezultujuće liste upisati u tekstulnu datoteku čiji je naziv treći argument komandne linije.

Napomena: U svakom redu izlazne datoteke treba upisati podatak o jednom krugu u obliku:

```
(x,y,r)
```

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

```
typedef struct stek {
  int niz[MAX];
  int tos;
} STEK;
```

i funkcije za dodavanje i brisanje sa steka:

```
int push(STEK *s, int podatak);
int pop(STEK *s, int *podatak);
```

Definisati funkciju koja na prvom steku *stek1* prepoznaje broj *n* (preuzimanjem sadržaja sa steka) i u novi stek *stek2*, koji je inicijalno prazan, upisuje brojeve koji predstavljaju koliko je broj *n* bio udaljen od originalnog vrha steka. Prototip funkcije je:

```
void prebacivanje(int n, STEK *stek1, STEK *stek2,
int (*push_function)(STEK*, int), int
(*pop_function)(STEK*, int*));
```

Upis na stek i preuzimanje sadržaja sa steka vršiti u sklopu funkcija koje su date kao argumenti prethodno navedene funkcije.

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

```
typedef struct graf
{
  int n;            // broj cvorova
  int info[MAX];    // inf.sadrzaj cvorova
  int ms[MAX][MAX]; // matrica susjednosti
} GRAF;
```

kojim se reprezentuje neusmjeren i povezan graf.

Neka je definisan tip:

```
typedef struct novi_graf
{
  int n;     // broj cvorova
  int *info; // inf.sadrzaj cvorova
  int **ms;  // matrica susjednosti
} NOVI_GRAF;
```

kojim se reprezentuje neusmjeren i povezan graf u kojem se informacioni sadržaj čvorova i matrica susjednosti dinamički alociraju.

Napisati funkciju sa promjenljivim brojem argumenata koja spaja proizvoljan broj grafova tipa GRAF, kreira i vraća NOVI_GRAF. Funkcija prihvata broj grafova *p*, te *p* pokazivača na grafove tipa GRAF. Spajanje vršiti tako da se zadnji čvor prethodnog grafa spaja sa prvim čvorom narednog grafa. U novom grafu alocirati tačno onoliko prostora koliko je potrebno za sve čvorove svih grafova tipa GRAF. Prototip funkcije je:

```
NOVI_GRAF *f(int p, ...);
```