Programiranje 2 — deseti par domačih nalog

(1) V datoteki `naloga1.h` so podane sledeče deklaracije tipov struktur:

```
typedef struct _PO {    // par predmet-ocena
    char predmet[4];    // enolična oznaka predmeta
    int ocena;          // ocena pri tem predmetu
} PO;

typedef struct _Student {    // podatki o študentu
    int vpisna;    // vpisna številka
    PO* po;        // kazalec na začetek tabele parov predmet-ocena
    int stPO;      // število parov predmet-ocena
} Student;

typedef struct _VO {    // par vpisna-ocena
    int vpisna;  // vpisna številka
    int ocena;   // ocena
} VO;
```

Datoteko `naloga1.c` dopolnite s sledečo funkcijo:

```
VO** opravili(Student** studentje, int stStudentov,
              char* predmet, int* stVO)
```

Funkcija naj vrne kazalec na začetek tabele kazalcev na strukture tipa `VO`, ki vsebujejo podatke o študentih, ki so opravili izpit pri predmetu z oznako `predmet`. Vsaka struktura naj vsebuje vpisno številko pripadajočega študenta in njegovo oceno pri omenjenem predmetu. Funkcija naj v spremenljivko, na katero kaže kazalec `stVO`, vpiše dolžino izdelane tabele. Strukture tipa `VO`, na katere kažejo kazalci v tabeli, naj bodo urejene po padajočih ocenah, v primeru enakih ocen pa po naraščajočih vpisnih številkah.

Parameter `studentje` je kazalec na začetek tabele kazalcev na strukture tipa `Student`, parameter `stStudentov` pa podaja dolžino te tabele. Študent opravi predmet, če pri njem prejme oceno najmanj 6. Lahko predpostavite, da ima pri istem predmetu vsak študent kvečjemu po eno oceno in da število parov predmet-ocena za vse študente skupaj ne presega 1000.

Dopolnite in oddajte datoteko `naloga1.c`. V paketu *Izhodiščne datoteke* na spletni učilnici najdete izhodiščno različico te datoteke, datoteko `naloga1.h` in en par testnih datotek.

(2) V datoteki `naloga2.h` so podane sledeče deklaracije tipov struktur:

```
typedef struct _Ulomek {     // ulomek
    int st;      // števec
    int im;      // imenovalec (> 0)
} Ulomek;

typedef struct _Tocka {      // točka (x, y)
    Ulomek x;    // koordinata x
    Ulomek y;    // koordinata y
} Tocka;

typedef struct _Premica {    // premica z enačbo y = kx + n oziroma x = n
    bool navpicna;    // true, če je navpična (premica ima enačbo x = n);
                      // false, če ni (premica ima enačbo y = kx + n)
    Ulomek k;     // parameter k (nima pomena, če je premica navpična)
    Ulomek n;     // parameter n
} Premica;
```

Datoteko `naloga2.c` dopolnite s funkcijo

`Tocka projekcija(Tocka t, Premica p)`

ki vrne pravokotno projekcijo točke `t` na premico `p`, torej presečišče premice `p` in tiste pravokotnice nanjo, ki poteka skozi točko `t`. V strukturi tipa `Tocka`, ki jo vrne funkcija, naj bodo ulomki okrajšani, imenovalci pa naj bodo pozitivni (ulomek $\frac{6}{-4}$, denimo, naj bo predstavljen kot $\frac{-3}{2}$, ulomek 0 pa kot $\frac{0}{1}$).

V testnih programih (datotekah `test*.c`) vsi števci pripadajo intervalu $[-30, 30]$, imenovalci pa intervalu $[1, 30]$.

Dopolnite in oddajte datoteko `naloga2.c`. V paketu *Izhodiščne datoteke* na spletni učilnici najdete izhodiščno različico te datoteke, datoteko `naloga2.h` in en par testnih datotek.