Strukture i pokazivači na strukture

1. NCP koji unosi sa standardnog ulaza unosi dva kompleksna broja (Realni, Imaginarni deo) i ispisuje na standardni izlaz njihov zbir, razliku, proizvod, kolicnik. Ako je drugi broj 0, onda prekinuti dalje učitavanje i ispis (deljenje nulom!!!) i ispisati vrednost imaginarne jedinice(0+i).

```
#include <math.h>  /*zbog pow(x,2)   */
#include <stdio.h>

typedef struct { double re, im; } Kompl;  /* Struktura kompleksnog broja. */

/*alternativa:  struct KB { double re, im; };  typedef struct KB Kompl ; */

Kompl  zbir    (Kompl a, Kompl b);
/*funkcija racuna a+b, kao rezultat vraca strukturu, tj. kompleksan broj */

Kompl  razlika (Kompl a, Kompl b);  /* a-b */

Kompl  proizvod (Kompl a, Kompl b);
/* funkcija racuna a*b, kao rezultat vraca strukturu, tj. kompleksan broj  */

Kompl  kolicnik (Kompl a, Kompl b) ; /*   a/b   */

main () {
  Kompl x, y, z;
   Kompl  j = {0, 1};
/*primer eksplicitnog zadavanja polja strukture Kompl: prvo polje re=0, drugo polje im=1 */


/*ucitavanje dva kompleksna broja - brojevi sa tastature se unose u polja strukture */
  while (1) {
    printf ("\nUnesite prvi broj x- Re, Im: "); scanf ("%lf%lf", &x.re, &x.im);
    printf (  "Unesite drugi broj y- Re, Im: "); scanf ("%lf%lf", &y.re, &y.im);
      if (y.re==0 && y.im==0) break;  /*bez  deljenja nulom */
    printf ("x     = (%f,%f)\n", x.re, x.im);
    printf ("y     = (%f,%f)\n", y.re, y.im);

    z = zbir(x, y);
    printf ("x+y   = (%f,%f)\n", z.re, z.im);

    z = razlika  (x, y);
    printf ("x-y   = (%f,%f)\n", z.re, z.im);

    z = proizvod (x, y);
    printf ("x*y   = (%f,%f)\n", z.re, z.im);

    z = kolicnik (x, y);
    printf ("x/y   = (%f,%f)\n", z.re, z.im);
  }

  z = proizvod (j, j);
   printf ("\nj^2   = (%f,%f)\n", z.re, z.im);
}
```

```
Kompl  zbir    (Kompl a, Kompl b)
{ a.re += b.re; a.im += b.im; return a; }

Kompl  razlika (Kompl a, Kompl b) { a.re -= b.re; a.im -= b.im; return a; }

Kompl  proizvod (Kompl a, Kompl b) { Kompl c;
  c.re = a.re * b.re - a.im * b.im; c.im = a.im * b.re + a.re * b.im;
  return c;
}

Kompl  kolicnik (Kompl a, Kompl b) { Kompl c;
  double d = pow(b.re, 2) + pow(b.im, 2);
  c.re = (a.re*b.re + a.im*b.im)/d; c.im = (a.im*b.re - a.re*b.im)/d;
  return c;
}
```

2. Napisati program koji učitava iz datoteke ulaz.txt informacije(naziv hale, ulica i grad, kapacitet izrazen u broju mesta za gledaoce, cenu renoviranja) o sportskim halama regiona (do 4 hale), a potom ispisuje na standardni izlaz sve rasplozive informacije za hale cija cena renoviranja ne prelazi 10000. Jednostavnosti radi, moze se pretpostaviti da se svi pojavni nazivi u ulaznoj datoteci sastoje od jedne reci (tj, mozete koristiti fscanf(...)).

ulaz.txt
Arena Savska Beograd 23000 10014.67
Spens Dunavska NoviSad 10000 9800.56
Zorka Drinska Šabac 8000 6700.43

```
#include <stdio.h>
#define MARKER_KRAJ_STRINGA   1
#define MAX_ZNAK              30
#define MAX_HALA              4

struct  hala_podatak {
  char  naziv [ MAX_ZNAK + MARKER_KRAJ_STRINGA ],  /*naziv hale */
            ulica[ MAX_ZNAK + MARKER_KRAJ_STRINGA  ],  /*naziv ulice hale */
            grad  [ MAX_ZNAK + MARKER_KRAJ_STRINGA  ]; /*ime grada kome pripada hala*/
  int   mesta;   /*kapacitet hale */
  float cena;    /*cena radova */
};

main( )
{
  struct hala_podatak NizHala[ MAX_HALA ]; /*niz struktura koji sadrzi podatke o halama */
  int i;  /*brojac u petlji */
  char s[6];  /* pomocna niska */
  float cenafix;
  FILE *f;

   f=fopen("ulaz.txt", "r");

/*ucitavanje podataka o svakoj hali iz datoteke ulaz.txt */
  for( i = 0; i < MAX_HALA; i++ )
{
```

```
  fscanf(f, "%s", NizHala[ i ].naziv);
  fscanf(f, "%s", NizHala[ i ].ulica);
  fscanf(f, "%s", NizHala[ i ].grad);
  fscanf(f, "%d",  &NizHala[ i ].mesta);
  fscanf(f, "%f",  &cenafix);

 NizHala[ i ].cena = cenafix;
 printf("\n\n");
 }



/*formatiran ispis hala cija cena renoviranja ne prelazi 10000*/
for( i = 0; i < MAX_HALA; i++ )
   if (NizHala[ i ].cena <= 10000)
     {  printf("\n");
        printf("==============HALA %d ================ "
          "\nNaziv:\t\t%25s"
          "\nUlica:\t\t%25s"
          "\nGrad:\t\t%25s"
          "\nKapacitet:\t%25d"
          "\nCenaRadova:\t\t%25.2f",
          i+1,
          NizHala[ i ].naziv,
          NizHala[ i ].ulica,
          NizHala[ i ].grad,
          NizHala[ i ].mesta,
          NizHala[ i ].cena);
  }

fclose(f);
return 0;
}
```

3. **NCP koji izracunava obim i povrsinu trougla i kvadrata.** Koristiti u program predstavljanje geometrijskih primitiva preko struktura.

```
#include <stdio.h>


/* Zbog funkcije sqrt. */
#include <math.h>

/* Tacke su predstavljene sa dve koordinate. Strukturom gradimo novi tip podataka. */
struct point
{int x;
int y;
};

/* Izracunava duzinu duzi zadatu sa dve tacke */
float segment_length(struct point A, struct point B)
{int dx = A.x - B.x;
  int dy = A.y - B.y;
return sqrt(dx*dx + dy*dy);
}
```

```
/* Izracunava povrsinu trougla Heronovim obrascem.
Argumenti funkcije su tri tacke koje predstavljaju temena trougla */

float Heron(struct point A, struct point B, struct point C)
{

/* Duzine stranica BC, AC, AB*/
float a = segment_length(B, C);
float b = segment_length(A, C);
float c = segment_length(A, B);

/* Poluobim */
float s = (a+b+c)/2;

/*Heronov obrazac */
return sqrt(s*(s-a)*(s-b)*(s-c));
}

/* Izracunava obim poligona. Argumenti funkcije su niz tacaka(niz struktura point)
koje predstavljaju temena poligona kao i njihov broj */
float circumference(struct point polygon[], int num)
{
int i; /*brojac u ciklusu */
float o = 0.0; /*obim */

/* Dodajemo duzine stranica koje spajaju susedna temena */
for (i = 0; i<num-1; i++) o += segment_length(polygon[i], polygon[i+1]);

/* Dodajemo duzinu stranice koja spaja prvo i poslednje teme */
o += segment_length(polygon[num-1], polygon[0]);

return o;
}

/* Izracunava povsinu konveksnog poligona. Argumenti funkcije su niz tacaka koje predstavljaju temena poligona
kao i njihov broj */
float area(struct point polygon[], int num)
{float a = 0.0; /* Povrsina */
int i; /* brojacka promenljiva */

/* Poligon delimo na trouglove i posebno izracunavamo povrsinu svakoga od njih */
for (i = 1; i < num -1; i++) a += Heron(polygon[0], polygon[i], polygon[i+1]);

return a;
}

main()
{
/* Definisemo dve promenljive tipa tacke */
struct point a;

/* Inicijalizujemo tacku b na (1,2) */
struct point b = {1, 2};

/* triangle je niz od tri tacke - trougao (0,0), (0,1), (1,0) */

struct point triangle[3];
```

```
/* square je niz od cetiri tacke - jedinicni kvadrat.
Obratiti paznju na nacin inicijalizacije niza struktura */
struct point square[4] = {{0, 0}, {0, 1}, {1, 1}, {1, 0}};

/* Postavljamo vrednosti koordinata tacke a*/
a.x = 0; a.y = 0;

/* Gradimo trougao (0,0), (0,1), (1,0) */
triangle[0].x = 0; triangle[0].y = 0;
triangle[1].x = 0; triangle[1].y = 1;
triangle[2].x = 1; triangle[2].y = 0;

/* Ispisujemo velicinu strukture tacka */
printf("sizeof(struct point) = %d\n", sizeof(struct point));

/* Ispisujemo vrednosti koordinata tacaka */
printf("x koordinata tacke a je %d\n", a.x);
printf("y koordinata tacke a je %d\n", a.y);
printf("x koordinata tacke b je %d\n", b.x);
printf("y koordinata tacke b je %d\n", b.y);

printf("Obim trougla je %f\n",circumference(triangle, 3));

printf("Obim kvadrata je %f\n",circumference(square, 4));

printf("Povrsina trougla je %f\n",Heron(triangle[0], triangle[1], triangle[2]));

/* Broj tacaka je moguce odrediti i putem sizeof */
printf("Povrsina kvadrata je %f\n",area(square, sizeof(square)/sizeof(struct point)));
}
```

Izlaz:

```
sizeof(struct point) = 8
x koordinata tacke a je 0
y koordinata tacke a je 0
x koordinata tacke b je 1
y koordinata tacke b je 2
Obim trougla je 3.414214
Obim kvadrata je 4.000000
Povrsina trougla je 0.500000
Povrsina kvadrata je 1.000000
```

4. NCP koji postavlja vrednosti koordinata tacke u 2-dimenzionoj ravni i ispisuje te koordinate. (Ilustracija koriscenja typedef.)

```
/* Koriscenje typedef radi lakseg rada */
#include <stdio.h>
#include <math.h>

/* Ovim se omogucava da se nadalje u programu umesto int moze koristiti ceo_broj */
typedef int ceo_broj ;



/* Ovim se omogucuje da se nadalje u programu umesto struct point moze koristiti POINT */
typedef struct point POINT;
```

```
struct point{
int x;
int y;
};

main()
{

/* Umesto int mozemo koristiti ceo_broj */
ceo_broj x = 3;

/* Definisemo promenljivu tipa tacke. Umesto struct point mozemo koristiti POINT */
POINT a;
printf("x = %d\n", x);

/* Postavljamo vrednosti koordinata tacke a*/
a.x = 1; a.y = 2;

/* Ispisujemo velicinu strukture tacka */
printf("sizeof(struct point) = %d\n", sizeof(POINT));

/* Ispisujemo vrednosti koordinata tacaka */
printf("x koordinata tacke a je %d\n", a.x);
printf("y koordinata tacke a je %d\n", a.y);

}
```

Izlaz:

```
x = 3
sizeof(struct point) = 8
x koordinata tacke a je 1
y koordinata tacke a je 2
```

5. NCP koji sa standardnog ulaza ucitava dva pravougaonika u ravni A, B i na standardni izlaz ispisuje povrsinu preseka pravougaonika, povrsinu unije pravougaonika, povrsinu razlike pravougaonika(A\B). Pravougaonik se zadaje preko x,y koordinata donjeg levog i gornjeg desnog temena pravougaonika. Stranice pravougaonika su paralelne koordinatnim osama.

Strukture se u funkcije cesto prenose po referenci. Moguce je koristiti pokazivace na strukture.

```
#include <stdio.h>
#include  <stdlib.h>

typedef   struct _rect
{ float  x,y;     /*koordinate donjeg levog ugla */
  float  x1,y1;   /*koordinate gornjeg desnog ugla */
} Pravougaonik ;

void ucitavanje (Pravougaonik *P);  /*ucitavanje donjeg levog i gornjeg desnog temena pravougaonika */

Pravougaonik  Presek(Pravougaonik *A, Pravougaonik *B);
/*presek pravougaonika A i pravougaonika  B */

float Povrsina (Pravougaonik *A);
```

```
/*izracunavanje povrsine datog pravougaonika */

main()
{   Pravougaonik  A,B ;          /*zadati pravougaonici */
    Pravougaonik presekAB ;   /*presek zadata dva  pravougaonika */

    float povrsinaA;  /*povrsina pravougaonika A*/
    float povrsinaB;  /*povrsina pravougaonika  B*/
    float povrsAB;    /*povrsina preseka pravougaonika A i pravougaonika B*/

   /*ucitavanja pravougaonika sa standardnog ulaza */
   ucitavanje(&A);
   ucitavanje(&B);


  /*odredjivanje A*B */
  presekAB  =  Presek  (&A  ,  &B);

  /*izracunavanje povrsine datih  pravougaonika i njihovog preseka */
 povrsinaA=Povrsina(&A);
 povrsinaB=Povrsina(&B);
 povrsAB=  Povrsina  (&presekAB);

 /*ispis rezultata */
printf( "Povrsina unije       :  %g\n", povrsinaA+povrsinaB-povrsAB);
printf( "Povrsina preseka     :  %g\n", povrsAB);
printf( "Povrsina prve razlike :  %g\n", povrsinaA - povrsAB);

return 0;
}

void ucitavanje (Pravougaonik *  P)
{  /*ucitavanje koordinata  za odgovarajuca dva temena */
   scanf("%f%f%f%f", &P->x,   &P->y,  &P->x1,    &P->y1);
}

Pravougaonik  Presek  (Pravougaonik *A, Pravougaonik  *B)
{
/*Racuna se i vraca presek pravougaonika A i B.
  Ako se A i B ne seku , funkcija vraca pravougaonik cije su sve koordinate jednake 0 */

 Pravougaonik  rezultat;   /*presek zadata dva pravougaonika */

/*odredjivanje koordinata donjeg levog ugla preseka-uociti tacka i strelica notaciju kod rezultat i kod A, B */
rezultat.x  = (A->x  > B->x) ? A->x : B->x;
rezultat.y  = (A->y  > B->y) ? A->y : B->y;

/*odredjivanje koordinata gornjeg  desnog ugla preseka => relacijski operator manje */
rezultat.x1  =     (A->x1  <  B->x1) ? A->x1 : B->x1;
rezultat.y1  =     (A->y1  < B->y1) ? A->y1 : B->y1;

/*pravljenje korekcije u slucaju da je rezultat preseka prazan */
  if  (rezultat.x  >=  rezultat. x1  ||  rezultat.y  >=  rezultat.y1  )
      rezultat.x=rezultat.y = rezultat.x1  =  rezultat.y1  = 0;
return rezultat;
}
```

```
float Povrsina  (Pravougaonik  *P )
{ /*vraca se rezultat mnozenja duzina dveju susednih stranica znajuci za paralelnost sa koordinatnim osama */
   return (P->x1 -P->x) * (P->y1 - P->y );
}
```

6. ***Strukture se u funkcije cesto prenose po referenci. Moguce je koristiti pokazivace na strukture.***
Šta je rezultat rada sledeceg programa?

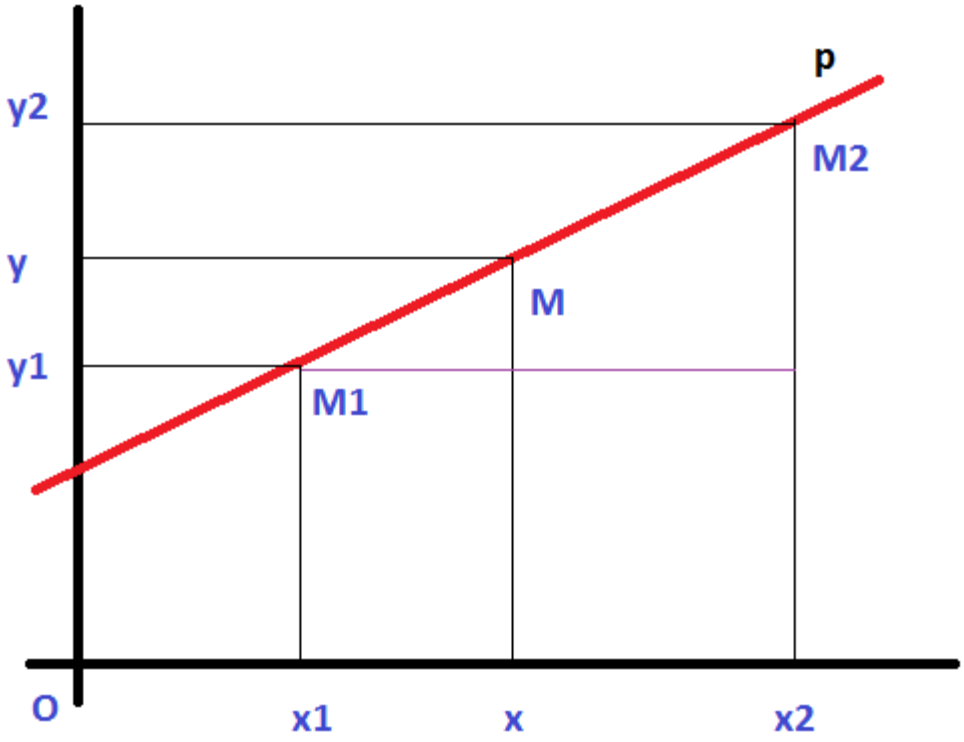
```
#include <stdio.h>
typedef struct point
{int x, y;
} POINT;

/* Zbog prenosa po vrednosti tacka ne moze biti ucitana */
void get_point_wrong(POINT p)
{
printf("x = ");
scanf("%d", &p.x);
printf("y = ");
scanf("%d", &p.y);
}

/* Koriscenjem prenosa preko pokazivaca, uspevamo */
void get_point(POINT* p)
{
/* p->x je skraceni zapis za (*p).x */
printf("x = ");
scanf("%d", &p->x);
printf("y = ");
scanf("%d", &p->y);
}

main()
{  POINT a = {0, 0};
   printf("get_point_wrong\n");
get_point_wrong(a);
printf("a: x = %d, y = %d\n", a.x, a.y);

printf("get_point\n");
get_point(&a);
printf("a: x = %d, y = %d\n", a.x, a.y);
}
```

7. **Sa standardnog ulaza se ucitava niz od n (n<100) tacaka u ravni takvih da nikoje tri tacke nisu kolinearne. Tacke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tacaka odredjuje konveksni mnogougao i rezultat ispisati na standardni izlaz.**

ULAZ
5
117 276
214 276
244 184
166 128
88 188

IZLAZ
Uneti mnogougao jeste konveksan!

```
#include<stdio.h>

typedef struct tacka
{ int x;  int y;
} TACKA;
```



(y2-y1):(y-y1)=(x2-x1):(x-x1) <=>
(x-x1)*(y2-y1)-(x2-x1)*(y-y1)=0
Funkcija po x,y: f(x,y)= (x-x1)*(y2-y1)-(x2-x1)*(y-y1)

Ako M(x,y) pripada pravoj ***p*** određenoj tačkama M1, M2, onda f(x,y)=0
Ako M(x,y) NE pripada pravoj ***p*** određenoj tačkama M1, M2, onda
  f(x,y)>0 za položaj tačke M s jedne strane prave ***p***
  f(x,y)<0 za položaj tačke M s druge strane prave ***p***

Dakle, ako su tačke T3(x3,y3), T4(x4,y4) sa različitih strana prave ***p,*** onda važi f(x3,y3)*f(x4,y4)<0

```
/* F-ja ispituje da li se tacke T3 i T4 nalaze sa iste strane prave odredjene tackama T1 i T2.*/
int SaIsteStranePrave(TACKA T1,TACKA T2, TACKA T3, TACKA T4)
{    int t3= (T3.x-T1.x)* (T2.y-T1.y) - (T2.x-T1.x)*(T3.y-T1.y); // T3 sa iste strane prave T1T2???
     int t4= (T4.x-T1.x)* (T2.y-T1.y) - (T2.x-T1.x)*(T4.y-T1.y); // T4 sa iste strane prave T1T2???
   return (t3 * t4 > 0);

}
int main()
{
   TACKA mnogougao[Nmax];
   int j,i;
```

```
   int n;
   int konveksan = 1;
//ucitavanje tacaka
   do
   {
      printf("Unesite broj temena mnogougla:\n");
      scanf("%d",&n);
      if(n<3) printf("Greska! Suvise malo tacaka! Pokusajte ponovo!\n");
   }
   while(n<3);

//ucitavanje tacaka mnogougla
   printf("Unesite koordinate temena mnogougla takve da nikoja tri temena nisu kolinearna!\n");
   for(i=0; i<n; i++) scanf("%d %d", &mnogougao[i].x, &mnogougao[i].y);

   /* Da bi mnogougao bio konveksan potrebno (i dovoljno) je da kada se povuce prava kroz bilo koja dva susedna
temena mnogougla sva ostala temena budu sa iste strane te prave.*/
   for(i=0; konveksan&&i<n-1; i++)
   {
      for(j=0; konveksan&&j<i-1; j++) konveksan=konveksan &&
SaIsteStranePrave(mnogougao[i],mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
      for(j=i+2; konveksan&&j<n-1; j++) konveksan=konveksan &&
SaIsteStranePrave(mnogougao[i],mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
      if(i!=0&&i!=n-1&&i+1!=0&&i+1!=n-1) konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
mnogougao[i+1],mnogougao[0],mnogougao[n-1]);
   }
   for(j=1; konveksan&&j<n-2; j++)
      konveksan=konveksan && SaIsteStranePrave(mnogougao[0],
     mnogougao[n-1],mnogougao[j],mnogougao[j+1]);

   if(konveksan) printf("Uneti mnogougao jeste konveksan!\n");
   else printf("Uneti mnogougao nije konveksan!\n");

   return 0;
}
```

8. **Sa standardnog ulaza se unose koordinate četiri tačke A, B, C i D (realni brojevi) koje pripadaju istoj ravni. Proveriti da li tačka D pripada ili ne pripada unutrasnjosti trougla ABC (tačke A, B, C nisu kolinearne) i dobijeni rezultat ispisati na standardni izlaz.**

Hint: Tačka D je u trouglu ako:
D, A je sa iste strane prave B, C &&
D, B je sa iste strane prave A, C &&
D, C je sa iste strane prave A, B

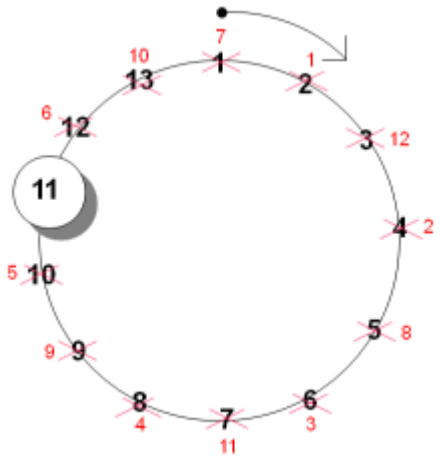
9. Opisati sve greške u sledećem C programu.

```
include <stdio.h>


define OFFSET 3;
define greska(msg) fprintf(stderr,#msg" %s\n",argv);
int length, width;
long area;
```

```
struct coord_p
{ int x,
int y }mypt


void f(void);

struct rectangle
{ coord_p topleft,
coord_p bottomrt } mybox

main(int argc, char argv**) {

if( argv = 3 ) /korektan broj argumenata /
{greska(Ne moze da se otvori)
exit(1)
} else
{ f();
}


return 3;
}


void f(void);
{boolean coord_test;
scanf("%d", mybox.topleft.x, mybox.topleft.y);
scanf("%d", mybox.bottomrt.x);

fscanf("%d", mybox.bottomrt.y);
coord_test=(bottomrt.x - topleft.x)*(bottomrt.y - topleft.y) > 0;
if (coord_test){ / izracunavanje duzine, visine i povrsine /
width = OFFSET+ bottomrt.x - topleft.x; length = OFFSET + bottomrt.y - topleft.y; Area = width * length;


printf("\nPovrsina je %l\n", area); }
}
```

10. Grupa od N ljudi redom numerisanih od 1 do N obrazovala je krug. Počevši od prvog čoveka, a krećući se u smeru kazaljke na satu iz kruga redom izlazi M-ti čovek. Preostali ljudi obrazuju novi manji krug pri čemu odbrojavanje počinje od levog suseda izbačenog. Celi brojevi M, N nisu unapred poznati, već se unose sa standardnog ulaza. NCP koji ispisuje redne brojeve osoba u redosledu napuštanja kruga.

/*Flavius Josephus priblem ili Josifov zadatak za N=13, M=2, redom izlaze: 2,4,6,8,10,12,1,5,9,13,7 ostaje: 11 */



```
#include <stdio.h>
#include <stdlib.h>

typedef struct covek_cvor
{ int rbr;
  struct covek_cvor *sledeci;
}covek;

main()
{ int i;       /*brojac u petljama */
  int  N;      /*broj ljudi u krugu */
  int M;       /*redni broj za izbacivanje */
covek *prvi,*p,*pom;
/*covek *p;
   covek *prvi;
   covek *pom;
*/

printf("Unesite N za ukupan broj ljudi koji obrazuje krug  i M za svaki M-ti koji se izbacuje iz kruga ");
scanf("%d %d",&N,&M);

/*prvi clan liste ljudi u krugu */
prvi=(covek*) malloc(sizeof(covek));
prvi->rbr=1;

p=prvi;
/*formiranje susednih clanova liste */
for(i=2;i < N+1;i++)
{ if (    (p->sledeci= (covek*) malloc(sizeof(covek) ) )  == NULL)
 { printf("Nema u memoriji dovoljno mesta za malloc\n");
   exit(1);
}
p=p->sledeci;
p->rbr=i;
}

/*zatvaranje kruga */
p->sledeci=prvi;

/*izbacivanje  M-tog  */
while( p != p->sledeci)   /*tj. dok ne ostane samo jedan u krugu */
{
for(i=1;i< M;i++)   p=p->sledeci;
printf("Izbaciti  %d\n", p->sledeci->rbr);
pom=p->sledeci;
p->sledeci=p->sledeci->sledeci;
free(pom);
}

/*ispis rezultata */
printf("I na kraju ostao je %d\n",p->rbr);
}
```

11. NCP koji iz neprazne datoteke broj.txt ucitava paran broj celih brojeva x[1], x[2],...,x[n], gde n nije unapred poznato. Ispisati poruku na standardni izlaz da li vazi da x[1]==x[n], x[2]==x[n-1],..., x[k]=x[k+1], k=1..n/2

```
#include <stdio.h>
#include <stdlib.h>

typedef struct listacv dvlista;
/*dvostruko povezana linearna lista*/
struct listacv
{  int broj; /*informaciono polje je clan niza x*/
   dvlista *sledeci, *prethodni;
};

dvlista *ubaci(dvlista *tek, int broj);
/*dodaje broju u clan liste tek i vraca cvor sa kraja liste*/

main()
{
 dvlista *prvi, *kraj;  /*prvi i poslednji cvor dvostruke liste*/
 int n;  /*broj ucitanih celih brojeva*/
 int broj; /*tekuci broj sa ulaza*/
 int jednak; /*indikator jednakosti dva clana - cuva vrednost 0 ili 1 */
 FILE *ulaz;
 int i;  /*brojacka promenljiva*/

 /*otvaranje datoteke za citanje i upis brojeva iz datoteke u
 dvostruku povezanu listu */
 ulaz=fopen("broj.txt", "r");
 fscanf(ulaz, "%d", &broj);

 /*kreiranje prvog cvora liste za prvi ucitan broj */
 prvi=(dvlista*)malloc(sizeof(dvlista));
 prvi->broj=broj;
 prvi->sledeci=NULL;
 prvi->prethodni=NULL;

 kraj=prvi; n=1; //za sada ucita je samo jedan clan

 /*formiranje liste od brojeva koji se ucitavaju sve do kraja datoteke - feof(ulaz) */
 while (!feof(ulaz))
 { fscanf(ulaz, "%d", &broj);
   kraj=ubaci (kraj, broj);
   n++;
 }

 fclose(ulaz);

 /*testiranje jednakih parova uz poeranje tekuceg prvog clana i tekuceg poslednjeg
 clana ka sredini liste*/
 jednak=1;
 for(i=1; i<=n/2 && jednak; i++)
 { jednak=jednak&&(prvi->broj==kraj->broj);
    prvi=prvi->sledeci;   /*pomeranje ka sredini liste */
             kraj=kraj->prethodni;  /*pomeranje ka sredini liste */
 }
printf("Za unete broje jednakost ");
if(!jednak) printf("ne ");
```

```
printf("vazi\n");
return 0;
}

dvlista *ubaci(dvlista *kraj, int broj)
{
  dvlista *novi;  /*novi cvor za umetanje u listu*/
  novi=(dvlista *)malloc(sizeof(dvlista)); /*alokacija memorije za novi cvor liste*/
  novi->broj=broj; /*inicijalizacija polja broj u cvoru novi */
  novi->sledeci=NULL;
  novi->prethodni=kraj; /*novi element se dodaje na kraj postojece liste*/
  kraj->sledeci=novi; /* do tada poslednji cvor kraj je ispred clana novi*/
  kraj=novi; /*poslednji dodat element je novi kraj liste*/
  return kraj;
}
```

12. Napisati program koji za dva polinoma stepena ne većeg od 20 unosi koeficijente (realni brojevi), a na standardni izlaz ispisuje:
   a) zbir ta dva polinoma
   b) vrednost oba polinoma u tački x, koja se zadaje kao argument komandne linije

Pretpostaviti da polinom je zadat strukturom

```
typedef struct polinom pol;
/*definicija polinoma - svaki polinom karakterisu stepen i koeficijenti */
struct polinom
{  int stepen;
   double koef[MAXEL];
};
```

13. Data je struktura

```
struct tacka{

int a;

int b;

char naziv[5];}
```

Ovom strukturom opisana je tacka sa koordinatama (a,b) u ravni kojoj je dodeljeno ima *naziv.* NCP koji ucitava dve tacke sa standardnog ulaza (ucitavaju se koordinate svake tacke i naziv) i ispisuje da li su date dve tacke jednake. Dve tacke su jednake ako su im iste obe koordinate.

14. Data je struktura

```
struct tacka{

int a;

int b;

char naziv[5];}
```

Ovom strukturom opisana je tacka sa koordinatama (a,b) u ravni kojoj je dodeljeno ima *naziv.* Napisati funkciju koja za dve promenljive tipa tacka kopira opis prve tacke u drugu promenljivu. NCP koji ucitava tacku sa standardnog ulaza (ucitavaju se koordinate tacke i naziv) i kopira je u drugu tacku.

15. Postoji popularna igra koja se igra u dve kategorije i svaki igrač ima neki broj poena u obe kategorije. Dobar igrač je svaki igrač za kog ne postoji neko ko je isti ili bolji u obe kategorije. Organizator će podeliti nagrade svim dobrim igračima a vi treba da nađete broj nagrada koje će biti podeljene.
Input: U prvom redu se nalazi broj igrača N, a u sledećih N redova se nalaze po dva broja x i y koji predstavljaju broj poena nekog igrača u dve kategorije, 0<= N <=100 000, 0<= x, y <=1 000 000
Output: Ispisati u prvom redu broj nagrada koje će biti podeljene.
Primer

Ulaz	Izlaz
7 1 8 3 4 2 3 2 5 10 1 6 2 5 1	5

Pojašnjenje: Treći (2,3) igrač nije dobar zbog drugog (3,4) i/ili četvrtog (2,5) igrača a sedmi (5,1) igrač nije dobar zbog petog(10,1) i/ili šestog igrača (6,2). Svi ostali igrači su dobri.

1. nacin - sortiranje niza struktura f-jom sort i posebnom funkcijom za poredjenje struktura

```
#include <cstdio>
#include <algorithm>

#define MAXN 100000

struct igrac{    int a,b;};

bool cmp (const igrac& x, const igrac& y)
   {       if (y.a != x.a) return x.a > y.a;
           return x.b > y.b;
   }

igrac niz[MAXN];  int n;

int main(){
   scanf("%d",&n);
   for(int i = 0; i < n; i++)
      scanf("%d %d",&niz[i].a,&niz[i].b);

   std::sort(niz,niz+n, cmp);

   niz[n].a = -1; niz[n].b = -1;
   int max1 = -1; //najveci broj poena u 2. kategoriji
   //u sortiranom nizu
   int rez = 0;
```

2. nacin – C++, sortiranje niza struktura f-jom sort i posebno redefinisanje operatora < za poredjenje struktura

```
#include <cstdio>
#include <algorithm>
#define MAXN 100000

struct igrac{
   int a,b;
   bool operator < (const igrac x)const
   {       if (a != x.a) return a > x.a;
       return b > x.b;
   }
};

igrac niz[MAXN];   int n;

int main()
{  scanf("%d",&n);
   for(int i = 0; i < n; i++)
scanf("%d %d",&niz[i].a,&niz[i].b);

  std::sort(niz,niz+n);
   niz[n].a = -1; niz[n].b = -1;
   int max1 = -1;
   int rez = 0;
   for(int i = 0; i < n; i++)
   {       if (max1 < niz[i].b && !(niz[i+1].a == niz[i].a
```

```
   for(int i = 0; i < n; i++)
   {
      if (max1 < niz[i].b &&
         !(niz[i+1].a == niz[i].a && niz[i].b ==
niz[i+1].b))        rez ++;

      max1 = (max1 < niz[i].b) ? niz[i].b : max1;
}

   printf("%d\n",rez);
   return 0;
}
```

```
&& niz[i].b == niz[i+1].b)) rez ++;

      max1 = (max1 < niz[i].b) ? niz[i].b : max1;
}

   printf("%d\n",rez);
}
```

3. nacin – upotrebom funkcije qsort iz stdlib.h

```
#include <stdio.h>
#include <stdlib.h>

typedef struct igraci{
 int prva;
 int druga;
} igraci;

void ucitaj(igraci *i, int n){

int j;
for(j=0; j<n; j++)
         scanf("%d %d", &i[j].prva, &i[j].druga);
}

void kopiraj(igraci *i,igraci *i1, int n){

int j;
for(j=0; j<n; j++){
         i1[j].prva = i[j].prva;
         i1[j].druga = i[j].druga;
}
}

void ispisi(igraci *i, int n){

int j;
for(j=0; j<n; j++)
         printf("%d %d\n", i[j].prva, i[j].druga);
}

int prvi(const void* v1,const void* v2){
    igraci *i1 = (igraci*) v1,*i2 = (igraci*) v2;
     if (i1->prva == i2->prva){
     if (i1->druga < i2->druga) return -1;
     else return 1;
     }
     else if (i1->prva < i2->prva) return -1;
     else return 1;


}

int drugi(const void* v1,const void* v2){
    igraci *i1 = (igraci*) v1,*i2 = (igraci*) v2;
     if (i1->druga == i2->druga){
     if (i1->prva < i2->prva) return -1;
     else return 1;
     }
     else if (i1->druga < i2->druga) return -1;
     else return 1;
```

```
}

int nije_jedinstven(igraci* i, int d, igraci i1, int k){
    if (k==0) return (i[k+1].prva == i1.prva && i[k+1].druga == i1.druga);
    else if (k==d-1) return (i[k-1].prva == i1.prva && i[k-1].druga == i1.druga);
    else return (i[k-1].prva == i1.prva && i[k-1].druga == i1.druga) || (i[k+1].prva ==
i1.prva && i[k+1].druga == i1.druga);
}

int main(){
    igraci *i;
    int n,j,k, br = 0;

    scanf("%d", &n);

    i = (igraci*) calloc(n, sizeof(igraci));

    ucitaj(i,n);
    qsort(i, n, sizeof(igraci), prvi);

    for(j=0; j<n-1; j++){
    if (nije_jedinstven(i,n,i[j],j)) continue;
     k = j+1;
     while(k<n && i[j].druga > i[k].druga){
     k++;
     }
     if (k == n) br++;
    }
    if (!nije_jedinstven(i,n,i[n-1],n-1)) br++;
//ispisi(i,n);

    printf("%d\n", br);
    free(i);
return 0;
}
```

16.

Perica igra jednu igru na svom računaru. On ima n svojih vojnika od kojih svaki ima neku jačinu. Dato je i n protivničkih vojnika od kojih svaki takođe ima neku jačinu. Jačine tih $2n$ vojnika su različite (tj. ne postoje dva vojnika sa jednakim jačinama). Perica treba da uradi sledeću stvar: treba da sastavi n parova vojnika tako da se svaki par sastoji od jednog njegovog i jednog protivničkog vojnika i da se svaki od $2n$ vojnika pojavljuje u tačno jednom paru. I tada kreće bitka. U svakom od n dvoboja (u i-tom dvoboju ($1 \leq i \leq n$) učestvuju vojnici i-tog para) pobeđuje vojnik koji je jači. Za svakog od n protivničkih vojnika data su po dva broja: jedan koji govori koliko Perica dobija poena ukoliko njegov (Peričin) vojnik pobedi tog vojnika i drugi koji govori koliko Perica gubi poena ukoliko njegov vojnik izgubi od tog vojnika. Perica na početku ima 0 poena. Odrediti koliki je maksimalan broj poena koji Perica može skupiti (taj broj može biti i negativan).

Ulaz.

(Ulazni podaci se nalaze u datoteci **vojnici.in**) U prvom redu tekstualne datoteke nalazi se prirodan broj n ($n \leq 2.000$). U drugom redu nalazi se n prirodnih brojeva: i-ti od tih brojeva ($1 \leq i \leq n$) predstavlja jačinu i-tog Peričinog vojnika (svaki od brojeva je manji od ili jednak 2.000.000.000). U trećem redu nalazi se $n$prirodnih brojeva: i-ti broj u tom redu ($1 \leq i \leq n$) predstavlja jačinu i-tog protivničkog vojnika (svaki od brojeva je manji ili jednak 2.000.000.000). U četvrtom redu nalazi se n prirodnih brojeva: i-ti broj ($1 \leq i \leq n$) predstavlja broj poena koji Perica dobija ukoliko je taj protivnički vojnik poražen (svaki od brojeva je manji od ili jednak 1.000). U petom redu nalazi se n prirodnih brojeva: i-ti broj ($1 \leq i \leq n$) predstavlja broj poena koji Perica gubi ukoliko je taj protivnički vojnik u dvoboju u kome je učestvovao izašao kao pobednik (svaki od brojeva je manji ili jednak 1.000).

Izlaz.

(Izlazne podatke upisati u datoteku **vojnici.out**) U prvom redu tekstualne datoteke ispisati jedan ceo broj a to je maksimalan broj poena koji Perica može skupiti.

Primer 1.

vojnici.in	**vojnici.out**
3 9 12 3 4 5 6 10 2 7 5 3 1	14

Resenje (bez struktura)

```
#include <stdio.h>
#include <stdlib.h>

int n;  int *A,*B,*D,*G;   int broj;

void zam(int *a,int *b)
{  int pom=*a;   *a=*b;  *b=pom;}

//prvo resenje ----------------------------------------
void sortA(int *A,int n)
{  int br=A[n/2];
   int i=0,j=n-1;

  while (i<=j)  {
    while (A[i]<br)      i++;
    while (A[j]>br)      j--;
    if (i<=j)    {      zam(A+i,A+j);      i++;      j--;}
}

  if (j>0)    sortA(A,j+1);
  if (n-i>1)    sortA(A+i,n-i);
}

void sortB(int *B,int *D,int *G,int n)
{  int br=B[n/2];
    int i=0,j=n-1;

  while (i<=j){
    while (B[i]<br)      i++;
    while (B[j]>br)      j--;
    if (i<=j)    {      zam(B+i,B+j);      zam(G+i,G+j);      zam(D+i,D+j);      i++;      j--;}
}

  if (j>0)    sortB(B,D,G,j+1);
  if (n-i>1)    sortB(B+i,D+i,G+i,n-i);
}

void f1()
{  sortA(A,n);
   sortB(B,D,G,n);

  int *P1=(int *)malloc(n*sizeof(int));
  int *P2=(int *)malloc(n*sizeof(int));
  int i,j;

  for (i=0;i<n;i++)
  {    int *pom=P2;
       P2=P1;
     P1=pom;
     if (A[i]>B[0]) P1[0]=D[0];   else      P1[0]=-G[0];

    for (j=1;j<=i;j++)
    {      P1[j]=P1[j-1]-G[j];
           if (A[i]>B[j] && P2[j-1]+D[j]>P1[j])  P1[j]=P2[j-1]+D[j];
```

```
   }
}

  broj=P1[n-1];
}

//-----------------------------------------------------
//drugo resenje ----------------------------------------
void sortC(int *B,int *D,int *G,int n)
{  int br=D[n/2]+G[n/2];
   int i=0,j=n-1;
 while (i<=j)  {
   while (D[i]+G[i]<br)      i++;
   while (D[j]+G[j]>br)      j--;
   if (i<=j)    {      zam(B+i,B+j);      zam(G+i,G+j);      zam(D+i,D+j);      i++;      j--;}
}
  if (j>0)    sortC(B,D,G,j+1);
  if (n-i>1)    sortC(B+i,D+i,G+i,n-i);
}

void f2()
{  sortC(B,D,G,n);
   broj=0;
   int i,j,da,k,m;
    m=n;
  for (i=n-1;i>=0;i--){

   da=0;
   for (j=0;j<m;j++)
     if (A[j]>B[i])
       if (!da)
       {         da=1;  k=j;}
       else          if (A[j]<A[k])  k=j;

   if (da)    {      m--;       zam(A+k,A+m);      broj+=D[i];}
   else      broj-=G[i];
}
}
//-----------------------------------------------------
void main()
{
  FILE *dat=fopen("Vojnici.in","r");
  fscanf(dat,"%d",&n);
  int i;
  A=(int *)malloc(n*sizeof(int));
  B=(int *)malloc(n*sizeof(int));
  D=(int *)malloc(n*sizeof(int));
  G=(int *)malloc(n*sizeof(int));
  for (i=0;i<n;i++)    fscanf(dat,"%d",A+i);
  for (i=0;i<n;i++)    fscanf(dat,"%d",B+i);
  for (i=0;i<n;i++)    fscanf(dat,"%d",D+i);
  for (i=0;i<n;i++)    fscanf(dat,"%d",G+i);
  fclose(dat);
  f1();
  //f2();
```

```
  dat=fopen("Vojnici.out","w");
  fprintf(dat,"%d",broj);
  fclose(dat);
}
```

Resenje 2 (sortiranje struktura)

```
#include <iostream>
#include <algorithm>
#include <stdio.h>

#define MAXVOJN 2000
#define MINJACINA -2000000001

using namespace std;

int n, perica[MAXVOJN],p,k,m1,s1;
struct PericaVojnik
{    int jacinaProtiv, dobija, gubi;} b[MAXVOJN];

bool cmp(PericaVojnik a, PericaVojnik b)
{    if(a.dobija+a.gubi>b.dobija+b.gubi)      return 0;
   return 1;
}

int main()
{    cin>>n;

   for(int i=0;i<n;i++)      scanf("%d", &perica[i]);
   for(int i=0;i<n;i++)       scanf("%d", &b[i].jacinaProtiv);
   for(int i=0;i<n;i++)       scanf("%d", &b[i].dobija);
   for(int i=0;i<n;i++)       scanf("%d", &b[i].gubi);
   sort(perica,perica+n);
   sort(b,b+n,cmp);
   int br=0;
   for(int i=n-1;i>=0;i--)
   {       int k=br;
      while(k<n&&perica[k]<b[i].jacinaProtiv)  k++;
      if(k>=n)
      {
         p-=b[i].gubi;
         br++;
      }
      else
      {
         p+=b[i].dobija;
         perica[k]=MINJACINA;
      }
   }

   cout<<p;
   return 0;
}
```