PROGRAMIRANJE 2 ABC, PROGRAMIRANJE II D
PRIPREMA ZA I KOLOKVIJUM (25 POENA)

POSTAVKE ZADATAKA

Izvor: Obnavljanje
Glavne teme: niska, argumenti komandne linije, rad sa datotekama.

1. Datoteka. Napisati program koji učitava sa komandne linije ime datoteke i realan broj x. Program treba da ispiše sve brojeve u datoteci koji su veći od x. Možete pretpostaviti da se u datoteci nalaze podaci tipa `float`. Koliko ima brojeva većih od x?

Napomena. Možete koristiti funkciju `atof()` za pretvaranje stringa u realan broj, radi x.

2. Sa standardnog ulaza se unose dvije niske koje predstavljaju elemente dva skupa. Elementi niski su mala slova. Skupovi nemaju više od 26 elemenata. Napisati program koji na standardni izlaz ispisuje niske koje predstavljaju: (a) uniju, (b) presjek i (c) simetričnu razliku elemenata dva skupa.

Primjer. Ako je $X_1 =$ "$abcxy$", $X_2 =$ "$axyz$" onda treba $X_1 \cup X_2 =$ "$abcxyz$" (ili neki drugi redosljed; redosljed nema značaja), $X_1 \cap X_2 =$ "axy", $X_1 \triangle X_2 =$ "bcz".

Po definiciji, $x \in Y \Leftrightarrow (x \in X_1 \wedge x \notin X_2) \vee (x \notin X_1 \wedge x \in X_2)$, gdje je $Y = X_1 \triangle X_2$, simetrična razlika.

3. Sa standardnog ulaza se unosi ime datoteke čija svaka linija sadrži po 6 cijelih brojeva: $x_1, y_1, x_2, y_2, x_3, y_3$ koji predstavljaju redom koordinate tjemena jednog trougla. Linija u datoteci nema više od 100. Napisati program koji će izbrojati koliko ima jednakokrakih, odnosno koliko ima pravouglih. Odgovarajuće rezultate štampati na ekran.

Dužina stranice trougla ili rastojanje dvije tačke u ravni iznosi $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, ako su u pitanju tačke (x_1, y_1) i (x_2, y_2).

Pravougli $a^2 + b^2 = c^2$.

4. Vrsta podataka `struct`. Pojedinoj tački u ravni pridruženi su njena boja i koordinate (x, y). Neka program sadrži deklaracije i instrukcije

```
struct { char color[10]; int x; int y; } A, B, C, point[6];
strcpy(A.color, "white"); A.x = 182; A.y = 243;
strcpy(B.color, "yellow"); B.x = -11; B.y = -12;
strcpy(C.color, "gray"); C.x = 87; C.y = 88;
```

Na početku rada programa, treba učitati podatke o $n = 6$ tačaka point[1], ..., point[6]. Za ilustraciju, ti podaci bi mogli da glase:

```
blue 2 1  blueberry 35 32  black 48 40
brown 77 75  green 0 -1  red -12 12
```

U nastavku rada, program treba da izračuna i saopšti sljedeće rezultate:

(a) Dužine stranica trougla ABC. Formula za dužinu stranice trougla ili rastojanje dvije tačke: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, ako su u pitanju tačke (x_1, y_1) i (x_2, y_2).

(b) Boju tačke koja je (među učitanim tačkama point[1], ..., point[n]) najbliža koordinatnom početku (0,0). Ako ima više najbližih tačaka point[i], onda saopštite boje point[i].color svih takvih tačaka.

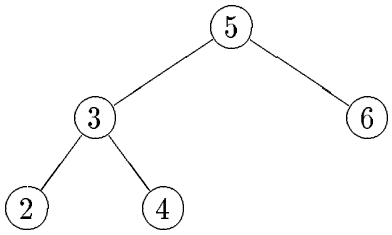
(c) Koordinate težišta sistema od učitanih tačaka point[1], ..., point[n], u oznaci (X, Y). Formule: $X = \frac{1}{n}\sum_{i=1}^{n} x_i$, $Y = \frac{1}{n}\sum_{i=1}^{n} y_i$, ako su tačke (x_i, y_i).

Izvor: Beograd, MATF, Programiranje 2

Glavne teme: pokazivač na strukturu, povezana lista, binarno stablo, rekurzija.

Binarna stabla

1. Napisati program koji za uređeno binarno stablo ispisuje elemente na najvećoj dubimi. Napisati funkcije za kreiranje čvora i unošenje elementa u stablo. Na primer, za stablo $(2 \leftarrow 3 \rightarrow 4) \leftarrow 5 \rightarrow 6$ program treba da ispiše: 2 4.



2. Napisati program koji za uređeno binarno stablo ispisuje sve listove (list je čvor stabla koji nema potomaka). Napisati funkcije za kreiranje čvora, unošenje elementa u stablo i ispis stabla. Na primer: za stablo $(2 \leftarrow 3 \rightarrow 4) \leftarrow 5 \rightarrow 6$ program treba da ispiše: 2 4 6 (isto stablo kao u prethodnom zadatku).

3. Napisati program koji za uređeno binarno stablo računa ukupan broj listova u stablu. Na primer: za stablo $(2 \leftarrow 3 \rightarrow 4) \leftarrow 5 \rightarrow 6$ program treba da ispiše: 3 (isto stablo kao u prethodnom zadatku).

4. Napisati program koji formira uređeno binarno stablo koje sadrži cele brojeve. Brojevi se unose sa standardnog ulaza sa nulom kao oznakom za kraj unosa. Napisati funkcije za:

- Ubacivanje elementa u ureď.eno stablo (bez ponavljanja elemenata);
- Ispisivanje stabla u inorder (infix) redosledu;
- Određ.ivanje najmanje dubine lista stabla.

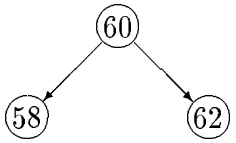
Na primer, za stablo: $(2 \leftarrow 3 \rightarrow 4) \leftarrow 5 \rightarrow 6$ funkcija treba da vrati 1 (jer se na toj dubini nalazi list 6) (isto stablo kao u prethodnom zadatku).

[U stablu, dubina vrha v definiše se kao dužina puta od korena stabla do tog vrha v. Znači, "rastojanje" od korena do v.]

5. Napisati program koji formira ureď.eno binarno stablo bez ponavljanja elemenata. Elementi stabla su celi brojevi i unose se sa standardnog ulaza (oznaka za kraj unosa je 0). Napisati funkcije za kreiranje elementa stabla, umetanje elementa u uređeno stablo, štampanje stabla i određ.ivanje sume elemenata u listovima stabla.

6. Napisati funkciju `int prebroj(cvor* drvo)` koja vraća broj elemenata stabla `drvo` koji su iste parnosti kao oba svoja sina (sva tri parna ili neparna). Ukoliko je čvor list ili ima samo jednog sina ne ulazi u zbir.

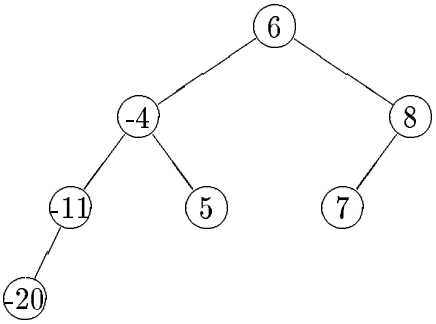
[U main–u, kreirati binarno stablo $58 \leftarrow 60 \rightarrow 62$ (prikazano je na slici), pozvati funkciju i štampati rezultat.



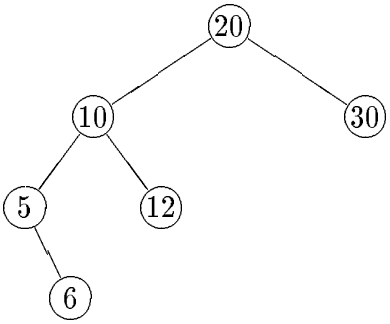
U našem slučaju, rezultat je 1.]

7. Napisati funkciju `int f5(cvor* drvo)` koja računa sumu svih elemenata drveta takvih da su veći od sume svojih direktnih potomaka. Ne računati čvorove koji nemaju ni jednog direktog potomka.

Na primer: drvo $((-20 \leftarrow -11) \leftarrow -4 \rightarrow 5) \leftarrow 6 \rightarrow (7 \leftarrow 8)$ vrednost funkcije -1.



8. Napisati funkciju `int f5(cvor* t, int d)` koja računa zbir svih parnih elemenata stabla t na dubini d, umanjen za zbir svih neparnih elemenata stabla t na dubini d. Na primer: ako $((5 \rightarrow 6) \leftarrow 10 \rightarrow 12) \leftarrow 20 \rightarrow 30$ i $d = 2$ onda je vrednost funkcije 7.



Povezane liste

9. Sa ulaza čitamo niz reči (dozvoljeno je ponavljanje reči) i smeštamo ih u listu (koja osim reči čuva i broj pojavljivanja za svaku reč). Napisati funkciju koja sortira listu algoritmom "bubble sort" po broju pojavljivanja reči. Napisati funkciju koja ispisuje listu počevši od reči koja se pojavila najviše puta.

10. Napisati program koji sa standardnog ulaza učitava pozitivne cele brojeve dok ne učita nulu kao oznaku za kraj. Na standardni izlaz ispisati koji broj se pojavio najviše puta među tim brojevima. Na primer, ako se na ulazu pojave brojevi: 2 5 12 4 5 2 3 12 15 5 6 6 5 program treba da vrati broj 5.

Zadatak rešiti korišćenjem dinamičke alokacije i strukture koja sadrži ceo broj i broj njegovih pojavljivanja.

11. Napisati program koji formira sortiranu listu od celih brojeva koji se unose sa standardnog ulaza. Oznaka za kraj unosa je 0. Napisati funkcije za:

- Formiranje čvora liste,
- Ubacivanje elementa u već sortiranu listu,
- Ispisivanje elemenata liste u rastućem poretku u vremenu $O(n)$,
- Ispisivanje elemenata liste u opadajućem poretku u vremenu $O(n)$.

Napomena: potrebno je da lista bude takva da funkcije za ispis liste u rastućem i u opadajućem poretku *ne koriste rekurziju niti dodatnu alociranu memoriju* a rade u vremenu $O(n)$.

12. Napisati program koji sa standardnog ulaza učitava tekst nepoznate dužine i smešta ga u (dinamički) niz karaktera. Oznaka za kraj unosa je uneta 0. Nakon toga ispisati uneti tekst na standardni izlaz po 10 karaktera u redu. Zadatak rešiti korišćenjem dinamičke alokacije.

Na primer, ako je uneto: "Ja polazem ispit iz programiranja2" program treba da ispiše:

```
Ja polazem
ispit iz p
rogramiran
ja2
```

13. Napisati program koji sa standardnog ulaza unosi sortirani niz celih brojeva dok ne unesemo nulu kao oznaku za kraj (niz alocirati dinamički). U slučaju da niz nije sortiran ispisati podatak o grešci na standardni izlaz a u suprotnom ispisati medijanu tog niza. Medijana (sortiranog) niza koji ima n članova je, u slučaju kada je n neparan broj, središnji element niza odnosno, u slučaju kada je n paran broj, srednja vrednost dva središnja elementa.

Na primer, ako je dat niz 1, 3, 5, 6, 12, medijana je broj 5 a ako je dat niz 3, 5, 8, 13, 20, 25, medijana je broj 10,5.

14. Napisati program koji radi sa tačkama. Tačka je predstavljena svojim x i y koordinatama (celi brojevi). Sa standardnog ulaza se učitava prvo broj tačaka a zatim koordinate tačaka. Dobijeni niz struktura sortirati pozivom funkcije "qsort". Niz sortirati po x koordinati, a ako neke dve tačke imaju istu x koordinatu onda ih sortirati po y koordinati.

Ako na ulazu dobijete niz: $(4,6)$, $(2,9)$, $(4,5)$, sortirani niz će izgledati: $(2,9)$, $(4,5)$, $(4,6)$.

15. Napisati funkciju `void zamene(CVOR** p)` koja menja povezanu listu koja je zadata svojim početkom tako da zameni mesta 1. i 2. elementu, potom 3. i 4. itd. Na primer, lista 1, 2, 3, 4 postaje 2, 1, 4, 3. Lista 1, 2, 3, 4, 5, 6 postaje 2, 1, 4, 3, 6, 5. Lista može sadržati i neparan broj elemenata, pri čemu u tom slučaju poslednji ostaje na svom mestu. Nije dozvoljeno formiranje nove liste – lista se može jedino preurediti u okviru postojećih struktura.

16. Napisati funkciju `Cvor* dodaj_u_listu(Cvor *glava, Cvor *novi, int n)` koja dodaje novi čvor na n–to mesto u listi. Ukoliko lista ima manje od n elemenata novi čvor se dodaje na kraj liste. Kao rezultat funkcija vraća adresu nove glave liste.

17. Napisati funkciju `void umetni(cvor* lista)` koja između svaka dva elementa u listi umeće element koji predstavlja razliku susedna dva.

18. Napisati funkciju koja iz date liste briše sledbenika prvog elementa u listi sa zadatom vrednošću, a ukoliko takav element ne postoji u listi, onda se briše prvi član liste.

19. Napisati funkciju `int f2(cvor* lista)` koja za elemente liste $a_1, a_2, \ldots, a_n$ računa $b = a_1 - a_2 + a_3 - \ldots + (-1)^{n+1} a_n$. Dozvoljeno je dodati još jedan argument u funkciji `f2`. Primeri: ako 4, 2, 5, 3, 8 onda $b = 12$; ako 12, 3, 3, 24, 25 onda $b = 13$: ako 4, 6, 2, 1, 2, 4 onda $b = -3$.

[U main–u, kreirati listu sa tri elementa $L = (10, 20, 30)$, pozvati funkciju i štampati rezultat b.]

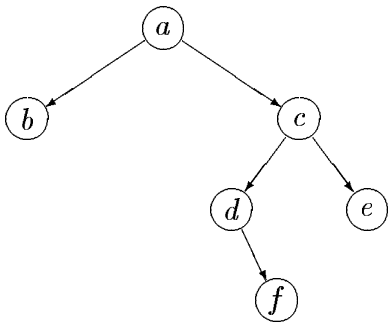
20. Napisati funkciju `void f3(cvor* lista, int k)` koja modifikuje zadatu listu tako što iza svakog broja djeljivog sa $k \geq 1$ umeće -1. Primeru: ako 4, 2, 5, 3, 8 i $k = 2$ onda 4, -1, 2, -1, 5, 3, 8, -1; ako 12, 3, 3, 24, 25 i $k = 3$ onda 12, -1, 3, -1, 3, -1, 24, -1, 25; ako 4, 6, 2, 1, 2, 4 i $k = 1$ onda 4, -1, 6, -1, 2, -1, 1, -1, 2, -1, 4, -1.

Izvor: Zagreb, MATH.HR, Strukture podataka i algoritmi

Binarna stabla

1. Napišite funkciju `void dvoje(BinaryTree T)` koja ispisuje oznake svih čvorova binarnog stabla koji imaju tačno dvoje djece. U primjeru dolje: a c.

Primjer: $b \leftarrow a \rightarrow ((d \rightarrow f) \leftarrow c \rightarrow e)$ (v. sliku).



2. Napišite funkciju `int dvoje(BinaryTree T)` koja vraća broj svih čvorova binarnog stabla koji imaju tačno dvoje djece. U primjeru gore: 2 (v. sliku).

3. Napišite funkciju `void dvoje(BinaryTree T)` koja u "postorder" redosljedu ispisuje oznake svih čvorova binarnog stabla koji imaju tačno dvoje djece. U primjeru gore: c a (v. sliku).

4. Napišite funkciju `void dubina(BinaryTree T, int k)` koja ispisuje oznake svih čvorova binarnog stabla na dubini k. U primjeru gore, za k = 2: d e (v. sliku).

5. Napišite funkciju `int dubina(BinaryTree T, int k)` koja vraća broj svih čvorova binarnog stabla na dubini k. U primjeru gore, za k = 3: 1 (v. sliku).

6. Napišite funkciju `node predak(BinaryTree T, node p, node q)` koja vraća onog zajedničkog pretka čvorova p i q koji ima najveću dubinu. U primjeru gore, za p = f i q = e: c (v. sliku).

Izvor: Wikipedia, the free encyclopedia

Binary tree

1. Za pojedini vrh v u binarnom stablu, njegov Strahlerov broj $s(v)$ definiše se na sljedeći način. Ako je v list onda $s(v) = 1$. Ako v ima samo jednog sina čiji je broj i onda $s(v) = i$. U slučaju da v ima dva sina, od kojih jedan ima broj $= i$ a drugi ima broj $< i$, stavlja se takođe $s(v) = i$. Najzad, ako v ima dva sina i oba imaju broj i (poklapa se), onda $s(v) = i + 1$. Napišite program na jeziku C u kome se, za dato binarno stablo, određuju i saopštavaju brojevi $s(v)$, za sve vrhove.

2. Razmotrimo binarno stablo $T = (V, E)$ i u njemu jedan vrh v. Bifurkacija $b(v)$ vrha v definiše se kao broj ivica / grana $e \in E$ u oblasti od v do listova. Napišite program na jeziku C u kome se, za dato binarno stablo T, određuju i saopštavaju vrijednosti $b(v)$ svih vrhova $v \in V$.

Dolaze dva zadatka

RJEŠENJA ZADATAKA

Zadaci iz obnavljanja

(1) Datoteka, koliko ima brojeva većih od x

Rješenje (proga.c):

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[]){
FILE* fp; float x, a; int n;
fp = fopen(argv[1], "r");
x = atof(argv[2]);
n = 0;
while (!feof(fp))
    {fscanf(fp, "%f", &a);
    if (a>x) printf("%f ", a);
    if (a>x) n = n+1;};
fclose(fp);
printf("\nkoliko %d\n", n);}
```

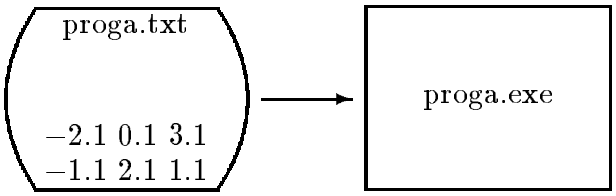
Pripremili smo datoteku proga.txt čiji sadržaj glasi:

```
-2.1 0.1 3.1
-1.1 2.1 1.1
```

Program se poziva sa argumentima, recimo:
proga proga.txt -0.5⟨enter⟩

Na ekranu će biti prikazani rezultati:

```
0.100000 3.100000 2.100000 1.100000
koliko 4
```



(2) Unija, presjek i simetrična razlika dva skupa (preko niski)

Rješenje (progb.c):

```
#include <stdio.h>
#include <string.h>
main(){ char x1[27], x2[27],
x3[27], x4[27], x5[27];
int I, J, K, M, N, P,
i, j, ima, m, n; char ch;
printf("Daj dva stringa (skupa)\n");
scanf("%s%s", x1, x2);
strcpy(x3, x1);
I = strlen(x1); J = strlen(x2);
K = strlen(x3);
for(j=0; j<J; j++)
    {ch = x2[j]; ima = 0;
    for(i=0; i<I; i++)
        {if (x1[i] == ch) ima = 1;
        if (x1[i] == ch) break;};
    if (!ima) x3[K] = ch;
    if (!ima) K = K+1;};
x3[K] = '\0';
K = 0;
for(i=0; i<I; i++)
    {ch = x1[i]; ima = 0;
    for(j=0; j<J; j++)
        {if (ch == x2[j]) ima = 1;
        if (ch == x2[j]) break;};
    if (ima) x4[K] = ch;
    if (ima) K = K+1;};
x4[K] = '\0';
M = strlen(x3); N = strlen(x4); P = 0;
for(m=0; m<M; m++)
    {ch = x3[m]; ima = 0;
    for(n=0; n<N; n++)
        {if (ch == x4[n]) ima = 1;
        if (ch == x4[n]) break;};
    if (!ima) x5[P] = ch;
    if (!ima) P = P+1;};
x5[P] = '\0';
printf("Unija %s (x3)\n", x3);
printf("Presjek %s (x4)\n", x4);
printf("Sym. dif. %s (x5)\n", x5);}
```

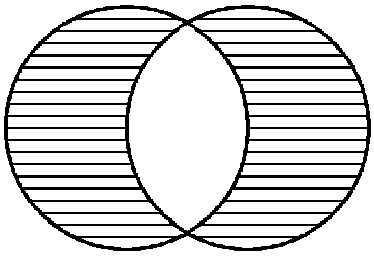
Prilikom propuštanja programa, na ekranu:

```
Daj dva stringa (skupa)
abcdxy axz
Unija abcdxyz (x3)
Presjek ax (x4)
Sym. dif. bcdyz (x5)
```

(od toga smo mi otucali abcdxy axz⟨enter⟩).

Komentar. U $x3 = x1 \cup x2$ uključuju se sva slova iz $x1$, a zatim i slova iz $x2$ kojih nema u $x1$. U $x4 = x1 \cap x2$ ona slova iz $x1$ koja se pojavljuju i u $x2$. U $x5 = x1 \triangle x2$ ona slova iz $x3$ kojih nema u $x4$. Naime, važi formula

$$x1 \triangle x2 = (x1 \cup x2) \setminus (x1 \cap x2), \quad \text{v. sliku.}$$



(3) Jednakokraki i pravougli trouglovi

Rješenje (progc.c):

```
#include <stdio.h>
#include <math.h>
main(){
FILE* fp; char filename[12];
int x1, y1, x2, y2, x3, y3, jk, pr;
long akv, bkv, ckv; float a, b, c;
int b1, b2;
printf("kako se zove datoteka\n");
scanf("%s", filename);
jk = 0; pr = 0;
fp = fopen(filename, "r");
    while (!feof(fp)){
fscanf(fp, "%d%d%d%d%d%d",
&x1, &y1, &x2, &y2, &x3, &y3);
akv=(x3-x2)*(x3-x2)+(y3-y2)*(y3-y2);
bkv=(x3-x1)*(x3-x1)+(y3-y1)*(y3-y1);
ckv=(x2-x1)*(x2-x1)+(y2-y1)*(y2-y1);
a = sqrt(akv);
b = sqrt(bkv);
c = sqrt(ckv);
b1 = a==b || a==c || b==c;
if (b1) jk = jk+1;
b2 = akv+bkv == ckv || akv+ckv == bkv
    || bkv+ckv == akv;
if (b2) pr = pr+1;};
fclose(fp);
printf("jednakokrakih %d\n", jk);
printf("pravouglih %d\n", pr);}
```

U tekućem folderu nalazi se datoteka pod nazivom progcdat.txt čiji sadržaj glasi:

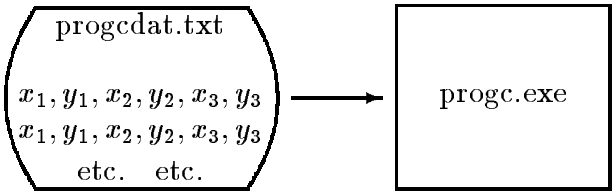
```
-2 0 2 0 0 10
0 0 4 0 0 3
0 0 7 0 0 7
0 0 4 7 11 22
```

Prilikom izvršavanja programa, na ekranu:

```
kako se zove datoteka
progcdat.txt
jednakokrakih 2
pravouglih 2
```

(od toga, mi progcdat.txt⟨enter⟩).

Komentar. Trougao je jednakokraki ako važi $a = b \vee a = c \vee b = c$. Trougao je pravougli ako važi $a^2 + b^2 = c^2 \vee a^2 + c^2 = b^2 \vee b^2 + c^2 = a^2$.



(4) Vrsta podataka "struct", tačke u boji

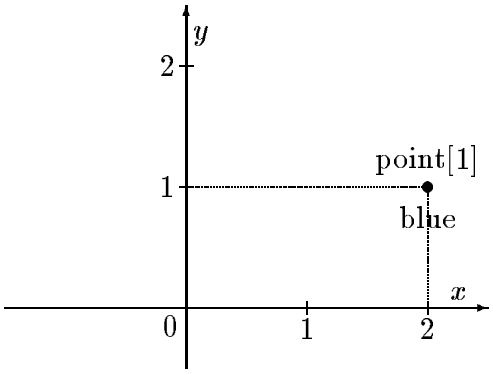
Rješenje (progd.c):

```
#include <stdio.h>
#include <string.h>
#include <math.h>
main(){
struct { char color[10]; int x;
    int y; } A, B, C, point[6];
int n, i, p, q;
float a, b, c, d[6], min, X, Y;
strcpy(A.color, "white");
A.x = 182; A.y = 243;
strcpy(B.color, "yellow");
B.x = -11; B.y = -12;
strcpy(C.color, "gray");
C.x = 87; C.y = 88;
n = 6;
printf("daj color,x,y puta %d\n", n);
    for(i=0; i<n; i++)
scanf("%s%d%d", point[i].color,
    &point[i].x, &point[i].y);
p = C.x - B.x; q = C.y - B.y;
a = sqrt(p*p + q*q);
p = C.x - A.x; q = C.y - A.y;
b = sqrt(p*p + q*q);
p = B.x - A.x; q = B.y - A.y;
c = sqrt(p*p + q*q);
printf("(a) stranice %f\n%f %f\n",
    a, b, c);
    for(i=0; i<n; i++)
```

```
{p = point[i].x; q = point[i].y;
d[i] = sqrt(p*p + q*q);};
min = d[0];
    for(i=1; i<n; i++)
if (d[i] < min) min = d[i];
    for(i=0; i<n; i++)
if (d[i] == min)
printf("(b) najbliza %s ",
    point[i].color);
X = 0; Y = 0;
for(i=0; i<n; i++)
{X = X+point[i].x; Y = Y+point[i].y;};
X = X/n; Y = Y/n;
printf("\n(c) teziste %f %f", X, Y);}
```

Prilikom rada programa, na ekranu:

```
daj color,x,y puta 6
blue 2 1  blueberry 35 32  black 48 40
brown 77 75  green 0 -1  red -12 12
(a) stranice 140.014282
181.796585 319.803070
(b) najbliza green
(c) teziste 25.000000 26.500000
```



(5) Dopuna o `malloc()`

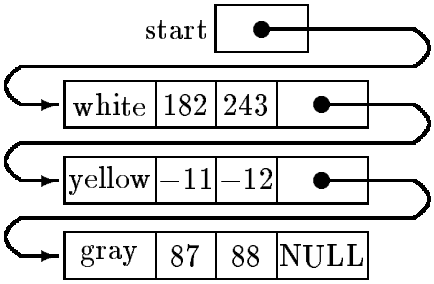
U prethodnom zadatku pojavljuje se niz tačaka A, B, C. Želimo da te tri tačke organizujemo u povezanu listu. Tada, dio programa može da glasi:

```
typedef struct tacka {char color[10];
    int x; int y; struct tacka *next;}
        tiptacka;
tiptacka *start, *A, *B, *C;
int duzina;
duzina = sizeof(tiptacka);
start = malloc(duzina); A = start;
strcpy(A->color, "white");
A->x = 182; A->y = 243;
A->next = malloc(duzina); B = A->next;
strcpy(B->color, "yellow");
B->x = -11; B->y = -12;
B->next = malloc(duzina); C = B->next;
strcpy(C->color, "gray");
C->x = 87; C->y = 88;
C->next = NULL;
```

Treba uključiti `string.h` zbog kopiranja. Takođe i `stdlib.h` zbog pozivanja `malloc()`. Svejedno je pisali `(*A).color` ili `A->color` (i ostalo).

Za promjenljivu `start` kaže se da predstavlja glavu liste. Sama lista sadrži tri elementa. Glava ne pripada listi. Ako `start == NULL` prazna lista. Na primjer, vidimo da je $*A = (\text{white}, 182, 243, B)$.

Vidimo da su promjenljive A, B, C suvišne. Umjesto njih, dovoljna je jedna promjenljiva, u oznaci recimo `t`, pokazivač na tekući element. Ona se uvodi u razmatranje preko deklaracije `tiptacka *t;` Zamislimo da želimo da pregledamo listu. Tada, na početku treba `t = start;` a tokom obilaska `t = t->next;` nekoliko puta. Kada `t == NULL` kraj liste.



Lako se utvrđuje da veličine pojedinih komponenti strukture iznose redom: 10 (niska), 4 (`int`), 4 (`int`) i 4 (pokazivač). Ukupno, jedna promjenljiva tipa `tiptacka` troši 24 bytes (prevodilac je dodao dva bajta).

Oslobađanje ili dealokacija prostora u memoriji tokom rada programa vrši se pomoću `free()`. Recimo, `free(C); B->next = NULL;` jedan element manje.