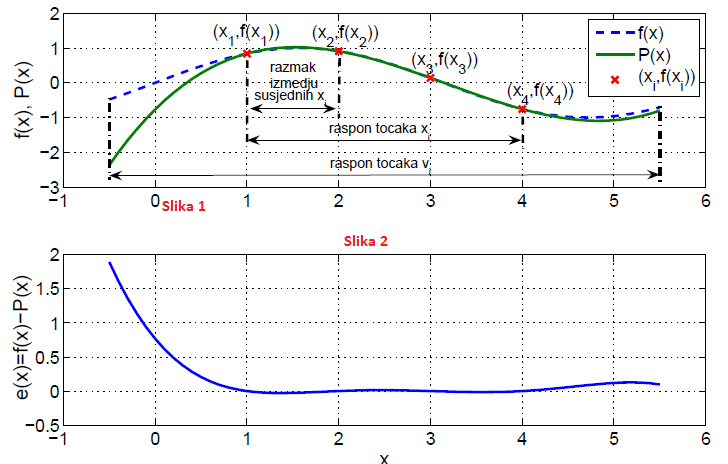
**MATLAB   
 DRUGA LABORATORIJSKA VJEŽBA   
(KODOVI, OBJAŠNJENJA, KOMENTARI I SLIKE IZ UPUTA)**  
  
  
Napomene: -ako već kopirate kodove, barem promijenite imena varijabli (skalara i vektora) i funkcija kako   
 sve ne bi bilo potpuno identično (npr. 'dx' umjesto 'lx', 'm' umjesto 'n' i slično…)  
 -u slučaju da vam treba objašnjenje značenja i način pozivanja određene funkcije (vjerojatno se,   
 u Matlabu, može namjestiti jezik), u 'Komandnom prozoru' Matlaba upišite: **help imefunkcije**   
 (npr. help polyfit)  
 -detaljnija objašnjenja mogu se pronaći i na stranici:   
 <http://www.mathworks.com/products/matlab/> , tj. upisivanjem **ključnariječ matlab**  
  
**1.a)**

 **KOD:**

x = 1:4; (-definiramo vektor 'x' u rasponu brojeva od 1 do 4 (primijeti dužinu 'raspon točaka x' na 'Slici 1')   
 -Matlab 'x' shvaća kao vektor, a mi ga doživljavamo kao os x)

y = sin(x); (definiranje funkcije koju ćemo aproksimirati)

k = min(abs(diff(x))); (-na 'Slici 1', na osi x, označene su točke 'xi'(-1, 0, 1, 2, 3, 4, 5 i 6), a te se točke zovu   
 ekvidistantne budući da je jednaka udaljenost između svake dvije susjedne točke  
 -ova linija koda, u varijablu 'k', pohranjuje 'najmanju apsolutnu vrijednost razmaka   
 (razlike)' između dvije ekvidistantne točke 'x' na osi x   
 (npr. oduzimanje točaka|3-2| = 1, |-1-0|=1 i slično, ali ne i npr. |4-2| = 2 pa da 2   
 bude najmanja apsolutna udaljenost između dviju ekvidistantnih točaka jer 2 to,   
 jednostavno, nije)

k = k / 10; (najmanji apsolutni razmak između dvije točke 'x' na osi x, kako zadatak kaže, moramo podijeliti   
 na deset jednakih dijelova i pohraniti u varijablu 'k', što činimo ovom linijom koda)

lx = abs(max(x) - min(x)); (u varijablu 'lx', pohranjujemo duljinu, tj. raspon vektora 'x' (označen   
 horizontalnom dužinom na 'Slici1') oduzimajući krajnju točku (x=4) i početnu   
 točku (x=1) vektora 'x' dobivajući rezultat lx=3)

v = (min(x)-lx/2):k:(max(x)+lx/2); (-na osi x, osim 'x-eva' definiranih vektorom 'x', moramo definirati dodatne   
 točke 'v' koje će, dakle, također biti na osi x, ali će njihov raspon, kako   
 zadatak kaže, morati biti dvostruko širi od raspona vektora 'x' (dakle,   
 raspon vektora 'v' = 2\*lx = 2\*3 = 6);   
 -vektor 'v' stvara se na način da se prepolovi raspon vektora 'x'   
 (lx/2 = 3/2 = 1.5) i taj 'poluraspon', vrijednosti 1.5, doda se lijevo od   
 minimalne točke vektora 'x', tj. x=1 (čime se dobiva početna točka   
 vektora 'v') te desno od maksimalne točke vektora 'x', tj. x=4 (čime se   
 dobiva krajnja točka vektora 'v')  
 -'k', u sredini, zapravo je vrijednosti k/10, a označava konstantni minimalni   
 razmak između svake dvije susjedne točke 'vi' vektora 'v'

yv = sin(v); (uz to što je y=sin(x) sinusoida gore definirana za 'x' točke vektora 'x', tako ćemo definirati i   
 novu funkciju yv=sin(v) za 'v' točke vektora 'v' budući da će se 'plotati' (crtati) i sinusoida i njena   
 aproksimacija, tj. dvije funkcije pa stvaramo 'zalihu' od te dvije funkcije koje će se crtati - 'y' i   
 'vy')

p = polyfit(x,y,length(x)-1); (-funkcija 'polyfit' računa koeficijente polinoma kojim će funkcija y=sin(x), ovdje   
 predana 'polyfitu' pomoću argumenata 'x' (varijabla, odnosno točke 'xi'   
 vektora 'x') i y (sama funkcija 'sin'), biti aproksimirana, a 'lenght(x)-1', kako   
 zadatak kaže, igra ulogu u tome da polinom bude 'm-1 reda'  
 -izračunato, 'polyfit' sprema u 'p', a imajte na umu da je 'p', u svojoj biti, vektor   
 koji sadrži koeficijente uz potencije 'x-eva' (u padajućem redoslijedu)   
 izračunatog aproksimirajućeg polinoma 'P', npr. p=[1 2 3 4 5];   
 -primjerice, 'polyfit' izračuna neki aproksimacijski polinom   
 P = **3**x^4 + **2**x^3 + **5**x^2 + **7**x^1 + **4**x^0 pa će vektor 'p' sadržavati koeficijente   
 uz potencije 'x-eva' padajućim redoslijedom, u ovom slučaju: p=[3 2 5 7 4]  
 -u slučaju da je P=**3**x^4 + **5**x^2 + **7**x + **4**, p neće biti p=[3 5 7 4], već p=[3 0 5 7 4]   
 -u padajućem redoslijedu, nedostaje x^3, ali samo zato što je koeficijent uz   
 njega 0 koji se ne smije izostaviti  
 -npr., za P=3x^4 + 2x^3 + 5x^2 + 7x, p=[3 2 5 7 0];   
 -u biti, ako je stupanj polinoma 'P' jednak broju n (najveća potencija uz x),   
 vektor 'p' će uvijek sadržavati n+1 članova)

f = polyval(p,v); ('polyval' je funkcija koja, pomoću predanih joj argumenata 'p' (koeficijenti polinoma koji   
 aproksimira funkciju, tj. konkretnu sinusoidu y=sin(x)) i točaka 'vi' vektora 'v' (u kojima se,   
 tj. za koje se vrši aproksimacija) računa vrijednost funkcije i sprema u 'f' što postaje   
 funkcija koja će se iscrtati kao polinom 'P')

subplot(2,1,1), plot(v,f,'-g', v,yv,'--b', x,y,'xr','MarkerSize',10);   
 (-zaredanim funkcijama 'subplot', može se crtati više podgrafova u istom Matlabovom 'plot'   
 okviru  
 -linija koja crta: **1.** v,f,'-g' (funkciju 'f' (koja je aproksimirajući polinom 'P') po varijabli 'v' (u   
 točkama 'vi' vektora 'v') punom zelenom linijom '-g'; uočite, na   
 'Slici 1', 'P' crtan punom zelenom linijom)

**2.** v,yv,'--b' (funkciju 'yv' (pravu, realnu sinusoidu) po varijabli 'v' (u   
 točkama 'vi' vektora 'v') crtkanom plavom linijom '—b')  
 **3.** x,y,'xr','MarkerSize',10 (u točkama 'xi'vektora 'x', za koje funkcija poprima   
 neku vrijednost, graf se obilježava markerom   
 (crvenim križićem) kako zadatak nalaže  
 -značenja brojeva, u (**2**,**1**,**1**), jesu slijedeća:   
 1.prvi broj (**2**): označava broj redaka slika u Matlabovom 'plot' (crtaćem) okviru

2.drugi broj (**1**): označava broj stupaca slika u Matlabovom 'plot' (crtaćem) okviru

3.treći broj (**1**): označava redni broj grafa (od ukupnog broja grafova) koji se crtaju u 'plot'   
 okviru)

subplot(2,1,1), ylabel ('f(x), P(x)'); (funkcija 'ylabel' dodjeljuje y osi neki naš proizvoljni naziv, a funkcija   
 'xlabel' dodjeljivala bi ime x osi)

subplot(2,1,1), grid; (funkcija 'grid', u koordinantni sustav, dodaje koordinantnu mrežu i brojčane oznake)

subplot(2,1,2), plot(v,yv-f,'b'); (funkcija 'plot', po točkama 'vi' vektora 'v' i plavom punom linijom 'b', crta   
 funkciju 'yv-f' koja je razlika funkcije prave sinusoide ('yv') i aproksimirane   
 sinusoide ('f') što je prikazano na 'Slici 2')

subplot(2,1,2), ylabel ('e(x)=f(x)-P(x)');

subplot(2,1,2), grid;

**1.b)**

Prije objašnjenja samog koda 1.b), nekoliko tehničkih napomena vezanih za neke definicije pojmova, pokretanja i spremanja.

Matlabov 'Workspace' ('Radna površina'), odnosno 'Komandni prozor', služi za upisivanje algoritama, definiranje varijabli, ali i za poziv funkcija. Komandni prozor je, kakve-takve usporedbe radi, neka vrsta 'main' funkcije koja poziva funkcije definirane 'negdje drugdje'.

'M-skripta' ('Matlab-Script') je vrsta Matlabove tekstualne datoteke u koju se mogu upisivati algoritmi i definirati varijable, ali se s tom tekstualnom datotekom, doduše, ne može raditi ništa 'korisno' (u smislu pokretanja i izvođenja), osim spremanja te skripte (s nastavkom .m) u neku arhivu na računalu odakle će se 'loadati' i nastaviti s pisanjem dotičnog algoritma.

'M-funkcija' ('Matlab-Function'), s druge strane, jest vrsta datoteke u kojoj se, nekim algoritmom i uvođenjem varijabli, može definirati funkcija koja obavlja određen posao nakon što se pozove u glavnom komandnom prostoru, a funkcija taj posao obavlja za određene argumente koji se, uz poziv funkcije, također proizvoljno unose u glavnom komandnom prozoru.   
  
Nova 'skripta', odnosno 'funkcija', stvaraju se preko 'New' --> 'Script' ili 'Function' u gornjem lijevom kutu.

Skripta se može napisati i spremiti bilo kako, bilo gdje, pod bilo kojim imenom te pokrenuti bilo kako i 'radit će', tj. otvorit će se.

S 'funkcijom' treba biti malo 'oprezniji'. Kod pisanja algoritma funkcije, prvi red svake funkcije, opisno, glasi:

**function argument-kojeg-funkcija-vraća = ime-funkcije (argument1, argument2, …)**

'Argument-kojeg-funkcija-vraća' i 'ime-funkcije', nazovite istim imenom radi neke 'sigurnost', da bi poziv radio ispravno, iako imena mogu biti različita. Stavite, npr.:

function **polinom** = **polinom**(tocke, crtanje)

Kad se funkcija kreira, stisne se 'Save', a funkcija se, kao vrsta jedne obične .m datoteke (npr. na 'Desktopu'), mora spremiti pod istim nazivom kakvim se naziva funkcija kako bi se poziv funkcije (u glavnom komandnom prozoru) dobro izvršio!   
(\*Ovdje, konkretno, funkcija se mora spremiti kao polinom.m)

Prije nego se funkcija pozove u komandnom prozoru, u dugačkoj bijeloj praznoj liniji, točno iznad radne površine, potrebno je postaviti dobru putanju ('path') do direktorija u kojem je korisnik, proizvoljno, spremio određenu funkciju polinom.m, te, konačno, odabrati tu funkciju u 'pathu'. Direktorij je tipa C:, D:, 'Local disk', 'Desktop' i slično (ne znam je li baš 'stručno' rečeno, ali poanta je tu…). Ako se ne postavi odgovarajući direktorij, poziv u komadnom prozoru javljat će grešku jer Matlab ne vidi direktorij u kojem se nalazi tražena funkcija.

Uzmimo, npr. funkciju **function polinom = polinom(tocke, crtanje)**

x = tocke;

…pri čemu je 'tocke' argument1, a 'crtanje' argument2

Ta funkcija, u komandnom prozoru, poziva se na sljedeći način:

**imefunkcije (argument1, argument2,…)**

Konkretno, za ovu funkciju, poziv glasi npr.:

**polinom ([1, 2, 3, 4, 5], 1)**   
…pri čemu je [1, 2, 3, 4, 5] argument1 ('tocke') unesen kao vektor što se od nas i očekuje, zbog prirode funkcije i zadatka, a '1' argument2 ('crtanje')

**KOD:**

function polinom = polinom(tocke, crtanje) (-posao ove funkcije 'polinom' (desna riječ 'polinom') je taj da,   
 pri njenom pozivu u komandnom prozoru (pri čemu joj se,   
 proizvoljno, daju argumenti 'tocke' i 'crtanje'), aproksimira   
 funkciju sinus i nacrta ju u 'plot' okviru;   
 -predajući 'tocke' kao vektor, mi funkciji 'polinom' predajemo   
 točke 'xi' u kojima želimo da se sinus aproksimira, a varijabla   
 'crtanje', dogovorno, može biti '0' ili '1' čime označavamo da   
 li želimo da se funkcija crta ili ne crta u 'plot' okviru (1=crtaj,   
 0= ne crtaj)

x = tocke;

y = sin(x);

k = min(abs(diff(x)));

k = k / 10;

lx = abs(max(x) - min(x));

v = (min(x)-lx/2):k:(max(x)+lx/2);

yv = sin(v);

p = polyfit(x,y,length(x)-1);

f = polyval(p,v);

if crtanje (ako smo unijeli 1 kao argument, crtaj funkciju, ako je unesena 0, nemoj ju crtati)

subplot(2,1,1), plot(v,f,'-g',v,yv,'--b',x,y,'xr','MarkerSize',10);

subplot(2,1,1), ylabel ('f(x), P(x)');

subplot(2,1,1), grid;

subplot(2,1,2), plot(v,yv-f,'b');

subplot(2,1,2), ylabel ('e(x)=f(x)-P(x)');

subplot(2,1,2), grid;

else disp('A-a, nema sinusoide');

end

end

**2.a)**

function perm = permutacija(niz, n) (-kako je naznačeno u jednom od prethodnih komentara o prototipu   
 funkcije, ovdje 'perm' označava izlazni argument funkcije, odnosno   
 ono što funkcija 'permutacija', nakon izvršavanja, vraća  
 -u sljedećem zadatku 2b), imat ćemo takav izlazni argument nazvan   
 'izlaz' koji će vraćati cijeli broj koji će označavati broj kojeg funkcija,   
 kako zadatak kaže, treba vratiti nakon izvršavanja)  
 -'niz' je vektor kojeg predajemo funkciji (npr. niz=[4 5 6]), a 'n' je cijeli   
 broj koji označava 'pokazivač' na jedan od članova u vektoru 'niz'   
 prema rednom broju  
 -npr. ako je n=1, 'n' će pokazivati na 1. član u vektoru niz, tj. na 4; ako   
 je n=3, 'n' pokazuje na 3. član u vektoru 'niz', tj. na 6  
 -'n' je neka vrsta 'ventila' - ako 'n' pokazuje na zadnji član polja, time   
 smo 'zatvorili' vektor 'niz' za daljnji kod koji bi, u tom vektoru 'niz',   
 morao pronaći sve moguće permutacije počevši od kombinacija od   
 prvog člana; što je 'n' manji (što je bliže pokazivanju na prvi član   
 vektora 'niz'), to će se više permutacija dobiti (najviše za n=1))

if n == length(niz) (ako je zadani 'n' jednak duljini vektora 'niz' (duljina vektora je definirana kao broj članova   
 vektora), pomoću 'n', kao ventilom, zatvorili smo vektor 'niz' (pjesnički rečeno) i jedina   
 permutacija je sam zadani vektor (u gore navedenom primjeru, za n=3 imat ćemo samo   
 permutaciju 456, dakle ništa kao 465, 654 i slično…)

disp(niz); (ispiši tu jedinu permutaciju)

else (ako zadani 'n' ne pokazuje na zadnji član vektora…)

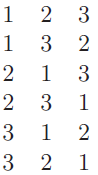
for i = n:length(niz) (iterator 'i' cijelih brojeva, postavi tamo gdje je 'n' i neka taj 'i' ide do zadnjeg člana   
 vektora 'niz', tj. do njegove konačne duljine)

tmp = niz(i); (-ove 3 linije koda obavljaju zamjenu vrijednosti među pojedinim članovima (pozicijama)   
 polja čime nastaju permutacije; konkretan primjer zamjene i nastanka jedne od   
 permutacija, priložen je na slikama ispod formalnog koda)  
 -sintaksa 'niz(i)', pri čemu je 'niz' vektor, a'i' neki cijeli broj, označava, tj. pristupa   
 određenom članu vektor 'niz' koji se nalazi na 'i-tom' rednom broju)

niz(i) = niz(n);

niz(n) = tmp;

permutacija(niz, n+1); (ovo je ponovni poziv funkcije 'permutacija', što znači da se ulazi u rekurziju, ali   
 argument više nije 'n', već 'n+1' čime se 'pokazivač' 'n' primiče zadnjem članu   
 polja i ide se prema kraju računanja preostalih permutacija)

niz(n) = niz(i); (brojni pozivi funkcije 'permutacija' gnijezde se jedan u drugog dok se ne dođe do kraja   
 kad zadnji 'podpoziv' nema više koga pozvati; tijekom povratka do prvog poziva funkcije   
 'permutacija', trebale bi se, u obrnutom poretku od željenog (kako je zadano u zadatku),   
 ispisivati zapamćene permutacije; ove 2 linije koda garantiraju ispis u traženom obliku:  
 

…a ne obrnuto 3 2 1

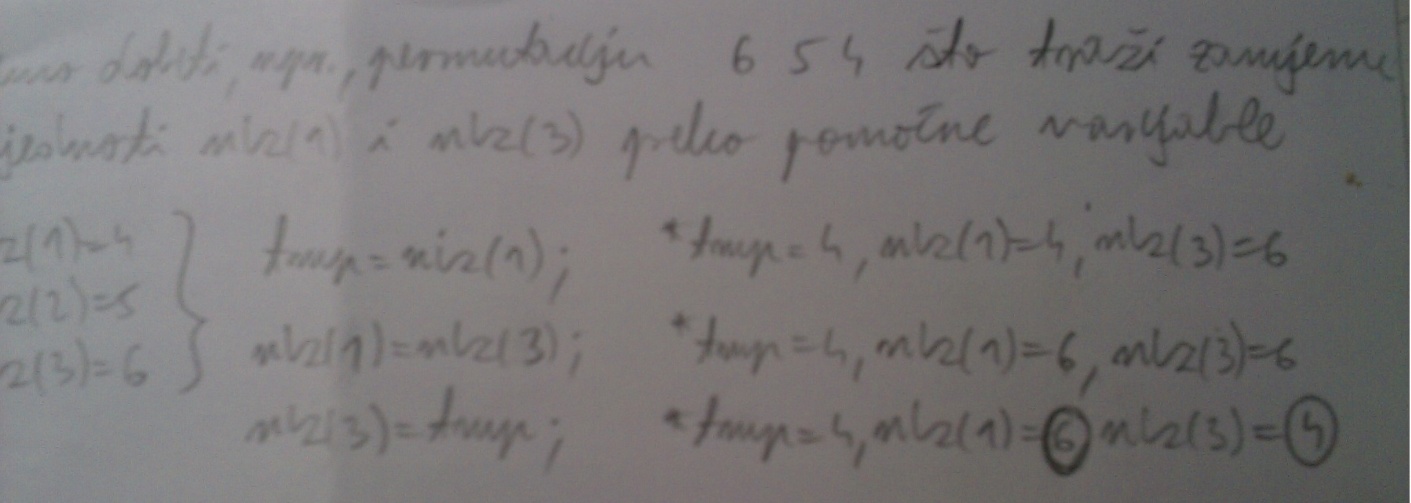
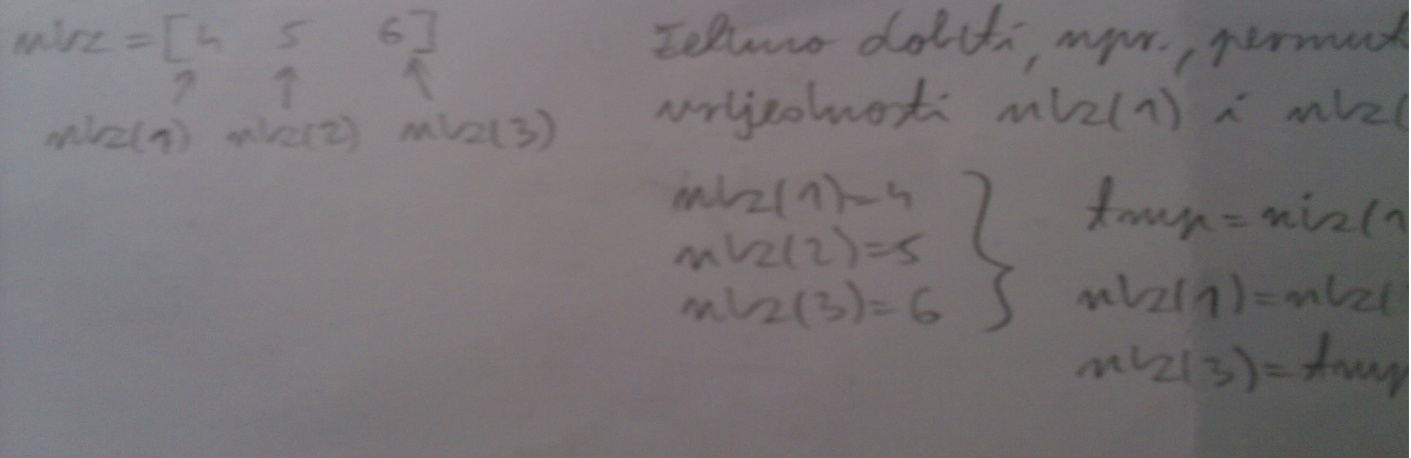
3 1 2

2 3 1 …

niz(i) = tmp;

end

end

end  
  
  
\*Primjer nastanka jedne od permutacija  


**2b)**

Bilo da je 'po defaultu', bilo da je korisnik ručno namjestio neki željeni 'path' (put) do određenog direktorija na računalu, Matlab, u svakom trenutku, vidi neki direktorij koji sadrži 'foldere' (mape), a svaka mapa 'file-ove' (datoteke). Sljedeći kod treba izbrojiti koliko foldera/fileova ima u direktoriju kojeg Matlab trenutno gleda (direktorij na kojeg je Matlab namješten, upućen) i vrati taj cijeli broj preko varijable, tj. izlaznog argumenta 'izlaz' (izlaznog argumenta kojeg svaka funkcija, prema ustaljenom prototipu, sadrži).

**KOD:**

function izlaz=ispis(x) ('x' je ulazni argument; u tekstu zadatka 2b), piše da je 'x' ime direktorija od kojeg   
 rekurzivni ispis počinje (polje znakova))

a = dir(x); (funkcija 'dir' prima 'x' (polje znakova) kao argument, prelistava sve foldere/fileove direktorija i   
 sprema ih u 'a'; time, 'a' postaje vektor čiji su članovi 'stringovi' (nizovi znakova), tj. imena   
 foldera/fileova)

n = length(a); (u varijablu 'n', funkcijom 'length', sprema se duljina (broj članova) vektora 'a')

for i = 1:n (iterira se ('broji') iterator 'i' po cijelim brojevima, od 1, do konačnog broja članova u vektoru 'a')

if a(i).isdir (funkcija 'isdir' provjerava je li trenutni 'i-ti' član vektora 'a' ('a(i)') folder ili nije; ako je folder,   
 ide se na sljedeću liniju koda)

if a(i).name ~= '.' (nažalost, bez detaljnijeg objašnjenja i prema saznanju od asistenta, '.' označava   
 (osigurava) da se ponovno ne prolazi direktorijem po istim pregledanim fileovima)

ispis([x,'\',a(i).name]); (po vektoru 'x', uzastopno se, po članovima, šeće rekurzijom (ponovnim   
 pozivom funkcije 'ispis'), ali argument nije više 'x', već [x,'\',a(i).name], pri   
 čemu je: -'x' = vektor foldera  
 -'\' = karakteristični ulazak u podređenu jedinicu, tj. iz foldera u   
 trenutni file 'a(i)' koji se, u rekurziji, šalje kao argument od   
 kojeg se nastavlja pretraga vektora imena 'x', svakim daljnjim   
 pozivom funkcije 'ispis', što osigurava da se, u direktoriju, ne   
 pretražuje po već pretraživanim mjestima)

izlaz = izlaz + ispis(x); (linija koja obavlja rekurzivan poziv funkcije 'ispis' i, svaki put za +1, povećava   
 'izlaz' (brojač članova vektora 'a') kojeg, u konačnici, treba vratiti kao rezultat)

end

else

disp(a(i).name); (funkcija 'disp' izvršava se ako, gore, 'isdir' utvrdi da 'a(i)' nije folder, te se ispisuje ime   
 filea)

izlaz = izlaz + ispis(x);

end

end

end