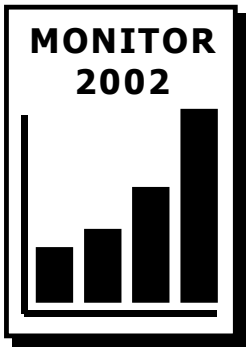
M O N I T O R 2002
pilotné testovanie maturantov



Informatika

2. časť

Odborný garant projektu: Štátny pedagogický ústav, Bratislava

Realizácia projektu: EXAM ®, Bratislava

01 *Medveď (10 bodov)*

Zo stanice TANAP-u vypustili po vyliečení medveďa. V snahe zistiť jeho zvyky, pripevnili mu vysielačku, ktorá každý deň v tú istú hodinu vysielala zmenu polohy medveďa (zmenu súradnice *x* a zmenu súradnice *y*) vzhľadom k pozícii z predchádzajúceho dňa. Údaje boli zaznamenávané do textového súboru *medved.txt* ako celočíselné hodnoty v metroch. V jednom riadku sú vždy dve čísla z jedného dňa. Stanica TANAP-u je na pozícii [0,0]. Prvý riadok v súbore je záznam z prvého dňa.

Doplňte pripravený program tak, aby vypočítal a vypísal

– koľko dní bol medveď sledovaný (premenná *pocet_dni*),
– v ktorý deň bol medveď najviac vzdialený od stanice TANAP-u (premenná *den_najdalej*),
– v ktorom dni sa medveď presunul o najdlhšiu vzdialenosť (premenná *den_najviac*),
– dĺžku trasy, ktorú medveď za celý čas prešiel (premenná *trasa*).

Riešenie všetkých štyroch úloh zapíšte do jediného cyklu. Pri riešení použite len jednoduché premenné.

```
Program Medvedik_putuje;
var x,y,pocet_dni,den_najdalej,den_najviac:integer;
    trasa:real;
    vstup:text;
    {DOPÍŠTE ďalšie potrebné deklaracie}



begin
 {otvorenie súboru}
  assign(vstup,'medved.txt');
  reset(vstup);
 {DOPÍŠTE  inicializovanie premenných}







  while not eof(vstup) do begin
    {DOPÍŠTE telo cyklu}
```

```
  end;
 {zatvorenie súboru}
  close(vstup);
  writeln('Medveď bol sledovaný ', pocet_dni, ' dní.');
  writeln('Najviac vzdialený bol ', den_najdalej, '. deň.');
  writeln('Najviac sa pesunul v ', den_najviac, '. deň.');
  writeln('Prešiel trasu dĺžky ', trasa, ' metrov.');
end.
```

02 *Hra s bodmi (20 bodov)*

V rovine je daných *n* bodov (*n* > 0), z ktorých žiadne tri neležia na jednej priamke. Body sú označené 1, 2, ..., *n*. Dvaja hráči s číslami 1 a 2 striedavo spájajú úsečkou dva nespojené body, každý svojou farbou. Prehráva hráč, ktorý svojou čiarou uzavrie trojuholník.

Súradnice bodov sú uložené v poli *sur*.
V dvojrozmernom poli *body* je uložený údaj o tom, ktoré body sú spojené úsečkou a ktorý hráč ich spojil. Ak body *i*, *j* sú nespojené, potom *body*[*i*, *j*] = 0, ak body *i*, *j* spojil hráč číslo 1, potom *body*[*i*, *j*] = 1, ak body *i*, *j* spojil hráč číslo 2, tak *body*[*i*, *j*] = 2.

Deklarácie premenných:

```
var sur:array[1..n] of record
                         x,y:integer;
                       end;
    body:array[1..n,1..n] of integer;
```

a) Napíšte procedúru *Zaciatok*, ktorá
- nakreslí na obrazovku všetkých *n* bodov ako kružnice so stredom v súradniciach bodu a polomerom 3,
- ku každému bodu napíše jeho číslo.

Predpokladajte, že
- súradnice bodov v poli *body* sú korektne zadané, všetky body sa zmestia na Vašu obrazovku,
- obrazovka je zmazaná a prepnutá v grafickom režime.

```
procedure Zaciatok;
  {DOPÍŠTE deklarácie premenných}



begin
  {DOPÍŠTE telo procedúry}









end;
```

Funkcia *Dobre* zisťuje, či v poli *body* pre každé *r*, *s*∈ {1, 2, ... ,*n*} platí *body*[*r*, *s*] = *body*[*s*, *r*].

```
function Dobre(n:integer):boolean;
var pokracuj:boolean;
    r,s:integer;
begin
  r:=1; s:=1;                              {1}
  pokracuj:=true;
  while pokracuj and (r<=n) do begin
    pokracuj:=body[r,s]=body[s,r];         {2}
    s:=s+1;
    if s>n then begin                      {3}
      r:=r+1; s:=1;                        {4}
    end;
  end;
  Dobre:=pokracuj;
end;
```

b) Napíšte prvých päť dvojíc prvkov poľa *body*, ktoré sa budú testovať v {2}, za predpokladu, že *n* = 3.

Odpoveď:

1. dvojica		
2. dvojica		
3. dvojica		
4. dvojica		
5. dvojica		

c) Pomocou *n* vyjadrite, koľkokrát sa vo funkcii *Dobre* vykoná príkaz {2}, ak výsledná hodnota funkcie je true.

Odpoveď: _____

d) Navrhnite, ako treba zmeniť príkazy v riadkoch {1}, {3}, {4} funkcie *Dobre* tak, aby sa netestovali prvky na diagonále a aby sa žiadna dvojica prvkov poľa netestovala v riadku {2} viackrát.

Odpoveď:

{1} _____

{3} _____

{4} _____

e) Napíšte funkciu *Tah*, ktorá
- zistí, či je ťah korektný (teda zadané body *bod1* a *bod2* sú z množiny {1, 2, ..., *n*}, a ešte nie sú spojené), ak je ťah korektný, výsledná hodnota funkcie je *true*, inak *false*,
- pri korektnom ťahu ho zaznamená do poľa *body*,
- nakreslí úsečku farbou hráča medzi spojenými bodmi.

```
function Tah(cislo_hraca:integer;bod1,bod2:integer):boolean;
  {DOPÍŠTE deklarácie premenných}




begin
  {DOPÍŠTE telo procedúry}









end;
```

Podľa pravidiel prehrá hráč, ktorý svojím ťahom vytvorí trojuholník. Procedúra *Trojuholnik* vypíše číslo hráča, ktorý prehral, ak spojením zadaných dvoch bodov vznikne trojuholník.

```
procedure Trojuholnik(bod1,bod2,hrac:integer);
 var bod3,pocet:integer;
begin
  if Tah(hrac,bod1,bod2) then
   begin
    pocet:=0;
   {1}
    for bod3:=1 to n do
     if     {2}
      then pocet:= pocet+1;
   {3}
    if pocet>0
     then OutTextXY(0,0,'Prehral hráč číslo '+     {4}     );
   end;
end;
```

f) Aká podmienka chýba na mieste {2} ?

Odpoveď: _____

g) Cyklus **`for`** medzi {1} a {3} sa vykoná *n*-krát. Opravte procedúru tak, aby cyklus skončil hneď po zistení prvého trojuholníka. Použite iný príkaz cyklu ako **`for`**.

Odpoveď:

h) Aký výraz treba doplniť na miesto {4}, aby procedúra vypísala číslo hráča, ktorý prehral?

Odpoveď: _____

03 *Bodový zisk (10 bodov)*

V poli *staty* máme uložené názvy štátov a ich bodový zisk za umiestnenia ich športovcov na olympiáde. V poli *poradie* sú indexy štátov podľa bodového zisku v zostupnom poradí. Teda na prvom mieste je index štátu, ktorý získal najviac bodov, na druhom mieste je index štátu s druhým najväčším počtom bodov atď. Pri rovnosti bodov na poradí nezáleží.

Príklad:

staty

	(1)	(2)	(3)	(4)	(5)
nazov	SR	ČR	USA	Poľsko	Rusko
body	20	20	300	50	300

poradie

	3	5	4	1	2

Definície a deklarácie:

```
type
 tStat = record
          nazov: string[15];
          pocet_bodov: integer;
         end;
{n je celočíselná konštanta, n ≥ 1}
 tStaty = array[1..n] of tStat;
 tPoradie = array[1..n] of integer;

var staty:tStaty;
    poradie:tPoradie
```

a) Procedúra *Vyhodnot* má vytvoriť pole *poradie* podľa zadania. Upravte procedúru *Vyhodnot* tak, aby do poľa *poradie* zapísala indexy všetkých štátov z poľa *staty* v požadovanej postupnosti, nielen do *poradie*[1]. Využite pripravenú procedúru, v napísanej časti urobte minimálne úpravy. Ak potrebujete, zadeklarujte ďalšie premenné.

```
procedure Vyhodnot(var poradie:tPoradie; staty:tStaty);
var max,imax,i:integer;




begin




  max:=-1;
  for i:=1 to n do
    if staty[i].pocet_bodov>max then
      begin imax:=i; max:=staty[i].pocet_bodov;end;
  poradie[1]:=imax;









end;
```

b) Doplňte procedúru *Vypis* tak, aby vytvorila binárny súbor s menom *meno_sub*. Binárny súbor bude obsahovať údaje z poľa *staty* usporiadané podľa bodového zisku. V prvej vete bude štát s najväčším bodovým ziskom a v poslednej bude štát s najmenším bodovým ziskom (pri rovnosti bodov na poradí nezáleží).

```
procedure Vypis(staty:tStaty; poradie:tPoradie, meno_sub:string);
var i:integer;
    subor:file of tStat;
begin
  {DOPÍŠTE otvorenie súboru}



 for i:=1 to n do
  begin
   {DOPÍŠTE}









  end;
  {DOPÍŠTE zatvorenie súboru}

end;
```

c) Niektoré štáty získali rovnaký počet bodov, ale nebrali sme to do úvahy. Procedúra *Spravodlivy_vypis* vypíše na začiatok riadka poradie a do toho istého riadka zoznam všetkých štátov, ktoré získali rovnaký počet bodov.

Príklad:

Poradie	štát
1.	USA Rusko
2.	Poľsko
3.	SR ČR

Doplňte procedúru *Spravodlivy_vypis*.

```
procedure Spravodlivy_vypis(staty:tStaty; poradie:tPoradie);
var i:integer;
  {DOPÍŠTE potrebné deklarácie}




begin
  {DOPÍŠTE}




  writeln('Poradie   štát');
  writeln('_____________________________________');
  for i:= 1 to n do begin
    {DOPÍŠTE}




  end;
end;
```

04 *Nájdi všetky výskyty (10 bodov)*

Pri práci so znakovými reťazcami môžeme používať štandardné funkcie **pos** a **copy**.

Funkcia **pos(podret,ret)** vráti index najľavejšieho výskytu podreťazca *podret* v danom reťazci *ret* alebo vráti 0, ak sa nevyskytuje.
Funkcia **copy(ret,zac,dlzka)** vráti podreťazec reťazca *ret* od pozície *zac* dĺžky *dlzka*.

Procedúra *Vyskyty* zisťuje výskyty podreťazca *podret* v reťazci *ret*. Indexy všetkých výskytov podreťazca v reťazci uloží do premennej *mnoz*.

Príklad:

V reťazci 'bababa jagababa' sa podreťazec 'aba' vyskytuje štyrikrát a preto v premennej *mnoz* bude po skončení procedúry *Vyskyty* hodnota [2,4,11,13].

Pomocou funkcií **pos** a **copy** dopíšte procedúru *Vyskyty*.

Definície a deklarácie:

```
type mnozina=set of 0..255;


procedure Vyskyty(podret,ret:string; var mnoz:mnozina);
var {DOPÍŠTE deklarácie premenných}



begin
  mnoz:=[];
  {DOPÍŠTE inicializáciu premenných}




  repeat
    {nájde nejaký výskyt podreťazca v reťazci DOPÍŠTE}



    {ak sa našiel, tak ho pridá do množiny}
    if   {DOPÍŠTE}



     then begin
       {DOPÍŠTE}




      end;
  until {DOPÍŠTE}

end;
```

05 ***Vyhoď zo zoznamu (10 bodov)***

Daný je jednosmerný spájaný zoznam.
Dopíšte procedúru *Vyhod*, ktorá z daného spájaného zoznamu vyhodí posledných *k* prvkov tak, že *k*-krát vyhodí posledný prvok (ak existuje) zo zoznamu. Aby sa mohol vyhodiť posledný prvok, treba nájsť predposledný a tomuto predposlednému zrušiť nasledovníka. Každý vyhadzovaný prvok zoznamu sa musí z pamäte správne uvoľniť (procedúrou **dispose**).

Definície a deklarácie:

```
type zoznam=^prvok;
     prvok=record
             info:real;
             dalsi:zoznam;
           end;


procedure Vyhod(var z:zoznam; k:integer);
  {DOPÍŠTE deklarácie lokálnych premenných}



begin
  while (z<>nil) and (k>0) do
    begin
      {ak je zoznam jednoprvkový}
      if  {DOPÍŠTE}

        then begin
          {zrušiť tento zoznam DOPÍŠTE}




        end
      else
        begin
          {nájde predposledný prvok a uvoľní jeho nasledovníka}




        end;
      dec(k);
    end;
end;
```

06 ***Rekurzia (10 bodov)***

Daná je rekurzívna procedúra *Rekurzia*.

```
procedure Rekurzia(n:integer);
 begin
  if n=0
   then write(n)
   else begin
         write(n);
         Rekurzia(n-1);
         write(n);
         Rekurzia(n-1);
         write(n);
        end;
 end;
```

a) Čo vypíše procedúra pre volania **`Rekurzia(1)`** a **`Rekurzia(2)`**?

Odpoveď:

Rekurzia(1) _____

Rekurzia(2) _____

b) Doplňte do tabuľky počet vypísaných znakov pre jednotlivé volania procedúry.

Volanie procedúry	Počet vypísaných znakov
`Rekurzia(1)`	
`Rekurzia(2)`	
`Rekurzia(3)`	
`Rekurzia(4)`	
`Rekurzia(5)`	

c) Aký je ciferný súčet čísla, ktoré vypíše procedúra po volaní **`Rekurzia(7)`** ?

Odpoveď: _____