OINARRIZKO PROGRAMAZIOA
1. azterketa partziala (2015-11-18)

Izen-deiturak: .....

Oharra: Azterketa partzial honen balioa azken notaren % 35 da, ebaluazio jarraituan. Azterketa gainditu egin behar da ebaluazio jarraituan jarraitu ahal izateko.

1. ariketa (0,8 puntu).

`P` prozeduraren eta `F` funtzioaren espezifikazioak kontuan hartuta, adieraz ezazu beheko sententzietako deiak zuzenak edo okerrak diren, zergatia arrazoituz. `X`, `Y` eta `Z` aldagaiak osoak dira, eta `R` boolearra.

```
procedure P (A, B : in Integer; C : in out Integer);
function F (M, N : in Integer) return Integer;
```

a) `P (3, 5, 7);`

Okerra: hirugarren parametroak aldagaia izan behar du deian, hirugarren parametro formala `in out` baita.

b) `P (X, Y);`

Okerra: hiru parametro behar dira deian.

c) `P (X, 2, Y);`

Zuzena.

d) `R := F (X, Z);`

Okerra: F funtzioak zenbaki osoa itzultzen du, eta ezin zaio, beraz, R boolearrari esleitu.

e) `X := F (Y, Z);`

Zuzena.

f) `P (X, Y, 1);`

Okerra: hirugarren parametroak aldagaia izan behar du deian, hirugarren parametro formala `in out` baita.

g) `Z := P (X, Y, Z);`

Okerra: P prozedura da, eta sententzia gisa egin behar zaio dei (deia ezin da espresio baten barruan egin).

h) `P (F (X, Y), 2, X);`

Zuzena.

2. ariketa (0,7 puntu).

Honako mota-definizio eta aldagai-erazagupen hauek emanik:

```
type Liburu is record
     Titulua: String (1 .. 20);
     Egilea : String (1 .. 20);
     Ale_Kopurua : Integer;
end record;

type Liburu_Bektore is array (1 .. 10) of Liburu;
type Taula is array (1 .. 10, 1 .. 10) of Boolean;
type Liburu_Lista is record
     Info : Liburu_Bektore;
     Zenbat : Integer;
end record;
```

```
Lib1, Lib2 : Liburu;
B1, B2 : Liburu_Bektore;
M : Taula;
X, Y : Integer;
P : Boolean;
L1, L2 : Liburu_Lista;
```

adieraz ezazu honako adierazpen hauek zuzenak edo okerrak diren, zergatia arrazoituz.

a) `X := Lib1.Egilea (1);`

Okerra: `X` osoari ezin zaio karaktere bat esleitu (`Lib1.Egilea (1)` string baten osagaia da, eta stringen osagaiak karaktereak dira).

b) `P := M (3, 8);`

Zuzena.

c) `P := M (3);`

Okerra: `M` matrizea da, bi dimentsioko array-a, eta, beraz, bi indize-espresio erabili behar dira osagai bat erreferentziatzeko.

d) `X := B1(3).Titulua;`

Okerra: `X` osoari ezin zaio string bat esleitu (`B1(3).Titulua` string bat da).

e) `X := B1(3).Titulua (2);`

Okerra: `X` osoari ezin zaio karaktere bat esleitu (`B1(3).Titulua (2)` string baten osagaia da, eta stringen osagaiak karaktereak dira).

f) `P := M (3) (3);`

Okerra: `M` matrizea da, bi dimentsioko array-a, eta, beraz, bi indize-espresio erabili behar dira osagai bat erreferentziatzeko; honela egin beharko litzateke: `M (3, 3)`.

g) `L1.Info(3).Titulua := L2.Info(5).Titulua;`

Zuzena.

3. ariketa (3 puntu).

Azpiprograma hauek emanik:

```
function Igarren_Digitua (N : in Integer; I : in Positive)
   return Natural;
   -- Aurrebaldintza: N, I (zenb. osoak), non I > 0
   -- Postbaldintza:
   --    emaitza = N-ren Igarren digitua

procedure Digituak_Kontatu (N : in Integer; Konta : out Natural);
   -- Aurrebaldintza: N (zenb. osoa)
   -- Postbaldintza:
   --    Konta = N-ren digitu kopurua
```

Espezifikatu eta inplementatu Adaz programa bat, zeinek, `0` zenbakiaz amaitutako zenbaki-sekuentzia bat irakurrita, zenbaki bakoitzeko nartzisista den ala ez idatziko baitu irteeran.

Esaten da n digituko zenbaki bat nartzisista dela, bere digituen n ordenako berreturen batura eta zenbakia bera berdinak direnean. **Zenbaki bat nartzisista den ala ez determinatzeko, inplementatu funtzio bat**.

Adibidez:

`153` nartzisista da, $1^3 + 5^3 + 3^3 = 153$ delako.

`1634` nartzisista da, $1^4 + 6^4 + 3^4 + 4^4 = 1634$ delako.

Oharra: Ariketa egiteko erabil ditzakezu `Irakurri_Osoa`, `Idatzi_Katea`... bezalako ohiko prozedurak.

Soluzioa:

```
function Nartzisista (N : Integer) return Boolean is
   Zenbat : Natural;
   Batura : Integer;
begin
   Batura := 0;
   Digituak_Kontatu (N, Zenbat);
   for I in 1 .. Zenbat loop
      Batura := Batura + Igarren_Digitua (N, I) ** Zenbat;
   end loop;
   return Batura = N;
end Nartzisista;

with Irakurri_Osoa, Idatzi_Katea;
procedure Zenbakiak_Nartzisistak_Direnetz_Aztertu is

   function Igarren_Digitua...;
   procedure Digituak_Kontatu...;
   function Nartzisista...;

   X : Integer;

begin
   Irakurri_Osoa (X);
   while X /= 0 loop
      if Nartzisista (X) then
         Idatzi_Katea ("Zenbakia nartzisista da");
      else
         Idatzi_Katea ("Zenbakia ez da nartzisista");
      end if;
      Irakurri_Osoa (X);
   end loop;
end Zenbakiak_Nartzisistak_Direnetz_Aztertu;
```

4. ariketa (3 puntu).

4.1. Honako konstante-definizio hauek emanik:

```
Hondar_Bektorea : constant array (0 .. 22) of Natural :=
   (2, 22, 17, 15, 19, 8, 10, 3, 0, 20, 1, 12, 6, 4, 21, 14, 5, 13,
    16, 18, 7, 11, 9);
Letra_Bektorea : constant array (0 .. 22) of Character :=
   ('W', 'E', 'V', 'S', 'L', 'P', 'X', 'A', 'T', 'C', 'R', 'N', 'Y',
    'G', 'K', 'Z', 'M', 'J', 'Q', 'H', 'F', 'B', 'D');
```

Espezifikatu eta inplementatu Adaz funtzio bat, zeinek, nortasun-agiri (NA) baten zenbakia emanda (zenbaki positiboa), dagokion letra kalkulatuko baitu. Horretarako, azpiprograman goiko bi konstante horiek definitu behar dira, eta letra kalkulatzeko erabili.

Honela egingo da kalkulua: NAren zenbakia 23rekin zatitu eta lortzen den hondarra hondar-bektorean bilatu; letra-bektorean posizio berean dagoena izango da NAren zenbakiari dagokion letra.

Adibidez:

`12345678` NAri dagokion letra `Z` da. Zenbakia `23`rekin zatitzen badugu, hondarra `14` da. Hondar-bektorean begiratzen badugu, `14` zenbakia `15`. posizioan dago (kontuan hartu indizea `0`tik hasten dela); letra-bektoreko `15`. posizioan `Z` letra dago.

`87654321` NAri dagokion letra `X` da (letra-bektoreko `6`. posizioan), `23`rekin zatituz gero hondarra `10` baita (hondar-bektoreko `6`. posizioan).

Soluzioa:

```
function Dagokion_Letra (DNI : in Integer) return Character is
   Hondar_Bektorea : constant array (0 .. 22) of Natural :=
      (2, 22, 17, 15, 19, 8, 10, 3, 0, 20, 1, 12, 6, 4, 21, 14, 5, 13,
      16, 18, 7, 11, 9);
   Letra_Bektorea : constant array (0 .. 22) of Character :=
      ('W', 'E', 'V', 'S', 'L', 'P', 'X', 'A', 'T', 'C', 'R', 'N', 'Y',
      'G', 'K', 'Z', 'M', 'J', 'Q', 'H', 'F', 'B', 'D');
   Hondarra, I : Integer;
   Aurkitua : Boolean := False;
begin
   I := 0;
   Hondarra := DNI mod 23;
   while I <= 22 and not Aurkitua loop
   -- kasu honetan, hondar posible guztiak bektorean
   -- daudenez, I <= 22 ken liteke begiztan jarraitzeko
   -- baldintzatik (ziur baikaude beti beteko dela)
      if Hondarra = Hondar_Bektorea (I) then
         Aurkitua := true;
      else
         I := I + 1;
      end if;
   end loop;
   return Letra_Bektorea (I);
end Dagokion_Letra;
```

4.2. Honako mota-definizio hauek emanik:

```
type NA_Bektore is array (1 .. 10) of Positive;
type NA_Letra_Bektore is array (1 .. 10) of Character;
```

Espezifikatu eta inplementatu Adaz prozedura bat, zeinek, 10 NAz osatutako bektore bat emanik (`NA_Bektore` motakoa), NA horiei dagozkien letrak kalkulatu eta beste bektore-batean (`NA_Letra_Bektore` motakoa) gordeko baititu. Horretarako, aurreko ataleko funtzioa erabili behar da.

Soluzioa:

```
procedure Dagozkien_Letrak_Kalkulatu
   (NA_Bektorea: in Na_Bektore; Letra_Bektorea : out NA_Letra_Bektore) is
begin
   for I in NA_Bektorea'First .. NA_Bektorea'Last loop
      Letra_Bektorea (I) := Dagokion_Letra (NA_Bektorea (I));
   end loop;
end Dagozkien_Letrak_Kalkulatu;
```

5. ariketa (2,5 puntu).

Mota-definizio hauek emanik:

```
type Taula is array (1 .. 20) of Integer;
type Lista is
   record
      Info : Taula;
      Zenbat : Natural;
end record;
```

eta honako `Ba_Dago` funtzio hau eginda dagoela suposatuz :

```
function Ba_Dago (L : in Lista; X : in Integer) return Boolean;
   -- Aurrebaldintza: L (zenbaki-lista estatikoa), X (zenbaki osoa)
   -- Postbaldintza:
   --    emaitza = True, baldin X L-n badago;
   --    emaitza = False, bestela.
```

Inplementatu, Adaz, hemen behean zehaztutako prozedura hau:

```
procedure Hirugarren_Lista_Eratu (L1, L2 : in Lista; L : out Lista);
   -- Aurrebaldintza: L1, L2 (zenbaki-lista estatikoak)
   -- Postbaldintza:
   --    L (zenbaki-lista estatikoa), honela eratuta:
   --      L1-eko e_i osagai (zenbaki) bakoitzeko:
   --        - e_i L2-n ere baldin badago, posizio berean,
   --          e_i bat egongo da L-n ere;
   --        - e_i L2-n ere baldin badago, baina beste posizio batean,
   --          0 bat egongo da L-n e_i horretarako;
   --        - e_i L2-n ez baldin badago, ez da ezer egongo L-n
   --          e_i horretarako.
```

Adibidez:

`L1 = (2, 3, 2, 6, 8)` eta `L2 = (2, 5, 3, 1, 8)` => `L = (2, 0, 0, 8)`
`L1 = (2, 3, 2, 6, 8)` eta `L2 = (2, 7, 2, 5, 6)` => `L = (2, 2, 0)`
`L1 = (2, 3, 6, 2)` eta `L2 = (2, 6)` => `L = (2, 0, 0)`

Soluzioa:

```
procedure Hirugarren_Lista_Eratu (L1, L2 : in Lista;
                                  L : out Lista) is
   -- Aurrebaldintza: L1, L2 (zenbaki-lista estatikoak)
   -- Postbaldintza:
   --    L (zenbaki-lista estatikoa), honela eratuta:
   --      L1-eko ei osagai bakoitzeko:
   --        - ei L2-n ere baldin badago, posizio berean,
   --          ei bat egongo da L-n ere;
   --        - ei L2-n ere baldin badago, baina beste posizio batean,
   --          0 bat egongo da L-n;
   --        - ei L2-n ez baldin badago, ez da ezer egongo L-n
   --          ei horretarako.
begin
   L.Zenbat := 0;
   for I in 1 .. L1.Zenbat loop
      if (I <= L2.Zenbat) and then (L1.Info (I) = L2.Info (I)) then
         L.Zenbat := L.Zenbat + 1;
         L.Info (L.Zenbat) := L1.Info (I);
      elsif Badago (L2, L1.Info(I)) then
            L.Zenbat := L.Zenbat + 1;
            L.Info (L.Zenbat) := 0;
      end if;
   end loop;
end Hirugarren_Lista_Eratu;
```