Estudis d'Informàtica, Multimèdia i Telecomunicació

Fonaments de Programació

PAC1 – Primera Prova d'Avaluació Continuada

Cognoms: *Escriu aquí els teus cognoms*

Nom: *Escriu aquí el teu nom*

Indicacions generals:

Raoneu i justifiqueu totes les respostes.

Les respostes incorrectes **no** disminueixen la nota.

Per a dubtes i aclariments sobre l'enunciat, adreceu-vos al consultor responsable de la vostra aula de teoria. Per a dubtes sobre codificació, adreceu-vos al consultor responsable de la vostra aula de laboratori.

Lliurament:

1. Cal lliurar la solució dels exercicis 1...10 en un fitxer anomenat *CognomsNom_FP_PAC1* adreçat a la bústia *"Lliurament d'activitats"* de l'aula de teoria.
2. Cal lliurar la codificació de la pregunta 10 a través del **Corrector Automàtic**, que trobareu a l'aula de teoria.
3. Data límit per lliurar la solució: **dilluns, 18 d'octubre de 2010** (a les 23:59 hores).

És imprescindible respectar el format i data d'entrega. La no adequació a aquestes especificacions pot suposar la no avaluació de la PAC.

Exercici 1: Identificació de tipus [10%]

Objectius: Mostrar coneixements elementals sobre els tipus, com manipular-los i identificar els tipus resultants d'efectuar operacions entre objectes elementals del llenguatge algorísmic

Materials: M1 i M2 fins M2.1

Tasca: Donades les següents definicions de tipus, constants i variables

```
tipus
        tBattery = {NiCd, NiMH, LiIon, LiSOCl2, LeadAcid}
        tSize = { D, C, AAA, AA}
ftipus

var
        voltage, power, energy : real;
        powerSource : tBattery;
        size: tSize;
        capacity :enter;
fvar
```

indicar quin es el tipus resultant d'avaluar les expressions següents:

i) integerToReal(capacity)*voltage > power

El resultat és de tipus **booleà**.

ii) capacity = realToInteger(power / voltage)

El resultat és de tipus **booleà**.

iii) realToInteger (energy) **div** capacity

El resultat és de tipus **enter**.

iv) size = AAA **i** powerSource = NiMH

El resultat és de tipus **booleà**.

Exercici 2: Sintaxi d'expressions [10%]

Objectius: Mostrar l'habilitat en la confecció d'expressions en llenguatge algorísmic. Utilització apropiada d'objectes elementals del llenguatge algorísmic per a la confecció de les expressions donades unes declaracions prèvies d'objectes elementals.

Materials: M1 i M2 fins M2.1

Tasca: Donades les següents definicions de tipus i variables

```
tipus
        tBattery = {NiCd, NiMH, LiIon, LiSOCl2, LeadAcid}
        tSize = { D, C, AAA, AA}
ftipus

var
        voltage, power, energy : real;
        powerSource : tBattery;
        size: tSize;
        capacity :enter;
fvar
```

Si les variables emmagatzemen dades relatives a una bateria en particular, escriure el llenguatge algorísmic corresponent a cada una de les descripcions següents:

i) Expressió que calculi quantes bateries serien necessàries per a disposar d'una capacitat de, com a mínim,10000 mAh, sabent que la variable *capacity* guarda la capacitat d'una sola bateria (el resultat final admet decimals).

```
10000.0 / integerToReal(capacity)
```

ii) Expressió per saber si la proporció entre *power* i *voltage* és superior a la proporció entre *energy* i *capacity*

```
( power / voltage ) > ( energy / integerToReal(capacity) )
```

iii) Expressió per saber si una bateria té un voltage superior a 3.5 V, és de tipus LiIon o LiSOCl2, i la seva grandària és D

```
(voltage>3.5) i (powerSource = LiIon o powerSource = LiSOCl2) i (size = D)
```

iv) Expressió per saber si una bateria és de grandària AAA, i la seva composició és NiCd.

```
(size = AAA)  i  (powerSource = NiCd)
```

Exercici 3: Avaluació d'expressions [10%]

Objectius: Mostrar l'habilitat en la manipulació d'expressions en llenguatge algorísmic. Donades unes declaracions prèvies d'objectes elementals, saber explicar-ne com efectuar la seva avaluació i donar-ne el resultat correcte.

Materials: M1 i M2 fins M2.1

Tasca: Donades les definicions de tipus i variables de l'exercici 2, i suposant que les variables tenen els valors següents:

voltage:=11.2, power:=100.0; energy := 25.7;
powerSource := LeadAcid;
size:=D;
capacity :=9850;

Calculeu el resultat de les expressions següents:

i) integerToReal(capacity)/1000.0 * voltage > power

9850.0/1000.0 * 11.2 > 100.0
9.850*11.2 > 100.0
110.32 > 100.0
CERT

ii) size = AAA **o** powerSource = NiMH **o no** capacity > realToInteger (voltage * energy)

FALS **o** FALS **o no** CERT
FALS **o** FALS **o** FALS
FALS

iii) integerToReal(capacity)/(voltage*power)<integerToReal(capacity **div** realToInteger(energy))

9850.0/(11.2*100.0)<integerToReal(9850 div 25)
9850.0/1120<394.0
8.79<394.0
CERT

iv) powerSource = LeadAcid **i** (**no** size = AA) **i** (energy * energy < power * voltage)

CERT **i no** FALS **i** 25.7*25.7 < 100.0*11.2
CERT **i** CERT **i** 660.49 < 1120.0
CERT **i** CERT **i** CERT
CERT

Exercici 4: Correctesa d'expressions [10%]

Objectius: Mostrar l'habilitat en la manipulació d'expressions en llenguatge algorísmic. Donades unes declaracions prèvies d'objectes elementals, raonar la correctesa de l'expressió, argumentant els motius d'incorrecció si n'hi ha.

Materials: M1 i M2 fins M2.1

Tasca: Donades les següents definicions de tipus, constants i variables

```
const
        MINDISTANCE: real = 10.0;
fconst

tipus
        tInterfaceWireless = {Bluetooth, Wifi, GSM, ZigBee, WiMax}
ftipus

var
        interface : tInterfaceWireless;
        range:real;
        inComputer: boolea;
        ratio: enter;
fvar
```

Digueu si són correctes sintàcticament o no les expressions següents. En cas que no ho siguin, especifiqueu-ne la raó o raons:

i) interface = Bluetooth **o** interface = Wifi **o** interface < GSM+WiMax

Incorrecte, no es poden fer operacions aritmètiques sobre tipus enumeratius.

ii) MINDISTANCE = integerToReal(range) / ratio

Incorrecte: no es pot fer l'operació / amb un enter (ratio) i a més range no és enter.

iii) inComputer = range > MINDISTANCE **o** interface = tInterfaceWireless

Incorrecte: no es pot fer una comparació amb un tipus de dades.

iv) Bluetooth<Wifi<GSM

Incorrecte: Els operadors de comparació donen com a resultat CERT o FALS i no es poden comparar amb altres tipus.

Exercici 5: Algorisme en llenguatge natural i especificacions [10%]

Objectius: Descriure un algorisme en paraules (llenguatge natural) i donar les especificacions (pre i postcondicions). Tan sols es pretén detectar l'habilitat en identificar els continguts dels diferents elements així com mostrar un coneixement bàsic de l'algorisme descrit.

Materials: M1 i M2 fins M2.2

Tasca: Es tracta de que descriviu i especifiqueu l'algorisme a seguir per tal de treure diners d'un caixer automàtic (evidentment no hi haurà solució única donat que diferents caixers segueixen diferents seqüències d'accions).

{pre: disposar d'un compte bancari, disposar d'una tarja de crèdit associada al compte, disposar de prou diners al compte, disposar d'un caixer automàtic amb prou diners en efectiu}

- Introduir la tarja al caixer automàtic
- Seleccionar l'opció per treure diners
- Introduir el codi secret
- Seleccionar l'import
- Demanar justificant de l'operació
- Esperar que el caixer faci l'operació
- Agafar la tarja
- Agafar els diners
- Agafar el justificant
- Guardar tarja, diners i justificant a la cartera
- Seleccionar no fer cap altra operació

{post: s'ha obtingut del caixer la quantitat de diners desitjada, quantitat que s'ha descomptat dels diners disponibles al compte, i de la quantitat de diners en efectiu del caixer.}

Exercici 6: Declarar les variables necessàries i donar les especificacions per a resoldre un problema [10%]

Objectius: Donada la descripció d'un problema, desclarar les variables necessàries i donar les especificacions (les precondicions i els postcondicions) de l'algorisme.

Materials: M1 i M2 fins M2.2

Tasca: Declareu les variables necessàries i doneu les especificacions d'un algorisme que resolgui el problema següent: "Es desitja calcular el temps mig per kilòmetre en minuts i segons emprat per un corredor de marató, donat el temps total en hores, minuts, i segons ". Considereu que la distància recorreguda és de 42,195 Km.

En concret, es demana que definiu les variables necessàries per representar les dades del problema (apartat var…fvar d'un algorisme) i que doneu la precondició i la postcondició.

```
var
        hoursM:       enter;
        minutesM:     enter;
        secondsM:     enter;
        minutesKm:    enter;
        secondsKm:    enter;
fvar

{Pre: hoursM = HOURS i minutesM = MINUTES  i seconsdM = SECONDS}

        CalcularTempsPerKm

{Post: minutesKm = realToInteger((hoursM*3600+minutesM*60+secondsM) / 42,195)
div 60 i  secondsKm = realToInteger((hoursM*3600+minutesM*60+secondsM) /
42,195) mod 60}
```

Exercici 7: Comprensió d'algorismes [10%]

Objectius: Avaluar coneixements bàsics d'algorísmica. Estructura general d'un algorisme i l'ús d'estructures algorísmiques bàsiques. Es plantegen preguntes d'interpretació donat un algorisme en llenguatge algorísmic.

Materials: M1 i M2 fins M2.3

Tasca: Considerant l'algorisme següent:

{Pre: n=N}

```
algorisme numero
var
    n,a,b: enter;
fvar

n:=readInteger();
b:=0;

mentre n≠0 fer
      a:=n mod 10;
      b:=(b*10)+a;
      n:=n div 10;
fmentre

writeInteger(b);

falgorisme
```

es demana:

i) Expliqueu què fa l'algorisme

Calcular els dígits del nombre d'entrada i crear un nou nombre amb els mateixos dígits colocats en ordre invers.

ii) En cas que dintre del bucle **mentre** canviem l'ordre de les instruccions de la següent manera:

```
b:=(b*10)+a;
a:=n mod 10;
n:=n div 10;
```

indiqueu com hauríem de modificar (si es que és necessari) l'algorisme per tal que continuï funcionant de la mateixa forma.

```
algorisme numero
var
    n,a,b: enter;
fvar

n:=readInteger();
b:=0;
a:=0;


mentre n≠0 fer
     b:=(b*10)+a;
     a:=n mod 10;
     n:=n div 10;

fmentre
b:=(b*10)+a;
writeInteger(b);

falgorisme
```

Exercici 8: Modificació del comportament d'un algorisme [10%]

Objectius: Avaluar coneixements bàsics d'algorísmica. L'avaluació va un pas més enllà de l'interpretació bàsica en plantejar modificacions del mateix. Es demanen modificacions al comportament d'un algorisme ja donat per tal de que aquest pugui contemplar situacions diferents.

Materials: M1 i M2 fins M2.3

Tasca: Considerant l'algorisme de l'exercici 7,

i) Com s'hauria de modificar l'algorisme per tal d'aplicar-lo a la part decimal d'un nombre real (la part fraccionària que va després del punt) de la mateixa manera que s'aplicava a un enter (L'algorisme només s'ha d'aplicar a aquesta part decimal, no a tot el nombre real)?

Solució – Extreure la part decimal i procedir com en el cas d'enters.

```
algorisme numero
var
    n,a,b: enter;
    r: real;
fvar

r:=readReal();
r:=r-integerToReal(realToInteger(r));{Es queda amb la part
decimal}
```

```
mentre r-integerToReal(realToInteger(r))≠ 0 fer
      r:= r * 10.0;
fmentre
n:=realToInteger(r);
b:=0;

mentre n≠0 fer
      a:=n mod 10;
      b:=(b*10)+a;
      n:=n div 10;
fmentre

writeInteger(b);

falgorisme
```

ii) Com s'hauria de modificar l'algorisme original per tal de considerar la possibilitat de que el nombre enter que es llegeix fós negatiu?

{Pre: n=N}

```
algorisme numero
var
    n,a,b: enter;
    sign: enter;
fvar
n:=readInteger();
si n<0 llavors
      sign := -1;
sino
      sign := 1;
fsi
b:=0;

mentre n≠0 fer
      a:=n mod 10;
      b:=(b*10)+a;
      n:=n div 10;
fmentre

b:= b * sign;

writeInteger(b);

falgorisme
```

Exercici 9: Bucles [10%]

Objectius: Avaluar coneixements d'algorísmica posant èmfasi en les composicions iteratives. Es demana reescriure una iteració emprant **per** ... **fer** en lloc de **mentre** ... **fmentre,** així com alguna petita modificació per tal de consolidar el coneixement que es té de l'algorisme reescrit.

Materials: M1 i M2 fins M2.3

Tasca: Considerant l'algorisme de l'exercici 7,

i) Reescriviu el mateix algorisme, però –aquest cop– utilitzeu l'estructura **per** ... **fper** en lloc del **mentre** ... **fmentre**, si és que és possible. Si no ho és, expliqueu-ne les raons.

No és possible fer la substitució perquè no coneixem el nombre de xifres de l'enter i per tant no sabem el límit d'un possible comptador per al bucle **per....fper**. Caldria fer servir primer un altre **mentre....fmentre** per obtenir dit nombre de xifres i consequentment l'algorisme resultant no seria eficient respecte a la versió inicial.

ii) Indiqueu totes les possibles ordenacions diferents de les sentències de dintre del **mentre...fmentre** que permetin que l'algorisme segueixi donant el mateix resultat sense haver de modificar res més.

```
...
mentre n≠0 fer
      a:=n mod 10;
      n:=n div 10;
      b:=(b*10)+a;
fmentre
...
```

Fixeu-vos que l'assignació `a:=n mod 10` cal executar-la abans de modificar el valor de la variable n, ja que l'expressió l'utilitza. I `b:=(b*10)+a` cal executar-la abans de modificar la variable *a* ja que l'utilitza.

Exercici 10: [10%]

Objectius: Donat un llistat en llenguatge algorísmic, traduir-lo a C. S'aconsella posar assignacions i comparacions, que en C requereixen utilitzar adientment el = i el ==.

Materials: M1 i M2 fins M2.3 així com la traducció de llenguatge algorísmic a C

Tasca: Es demana que codifiqueu l'algorisme següent en llenguatge C, el compileu i el proveu. Els resultats es deuen mostrar amb dos decimals.

{Pre: A l'entrada estàndard hi ha una seqüència de la forma: `x1 y1 x2 y2` , que representen les coordenades inicials i finals d'una línia de pendent menor de 45º, respectivament.}

```
algorisme coordenades
var
      dx, dy, p, end: enter;
      x1, x2, y1, y2, x, y: real;

      x1 := readReal();
      y1 := readReal();
      x2 := readReal();
      y2 := readReal();

      si x1>=x2 llavors
            dx := realToInteger(x1 - x2);
      sino
            dx := realToInteger(x2 - x1);
      fsi

      si y1>=y2 llavors
            dy := realToInteger(y1 - y2);
      sino
            dy := realToInteger(y2 - y1);
      fsi

      p := 2 * dy - dx;

      si x1 > x2 llavors
            x := x2;
            y := y2;
            end := realToInteger(x1);
      sino
            x := x1;
            y := y1;
            end := realToInteger(x2);
      fsi

      writeReal(x);
      writeReal(y);


      mentre x < integerToReal(end) fer
            x := x + 1.0;
            si p < 0 llavors
                  p := p + 2 * dy;
```

```
            sino
                  y := y + 1.0;
                  p := p + 2 * (dy - dx);
            fsi
            writeReal(x);
            writeReal(y);
      fmentre
falgorisme
```

{Post: S'ha escrit per la sortida estàndard la seqüència que correspon a la llista de punts `(x,y)` que hi ha en la línia entre `(x1,y1)` i `(x2,y2)`.}

Per exemple, si l'entrada de coordenades correspon als valors 10.0 10.0 20.0 15.0, la sortida correspondrà als punts (escrits en columna per a millorar la seva legibilitat, no és necessari fer-ho així al codi resultant):

10.00 10.00
11.00 11.00
12.00 11.00
13.00 12.00
14.00 12.00
15.00 13.00
16.00 13.00
17.00 14.00
18.00 14.00
19.00 15.00
20.00 15.00

```
/* PAC1: Algorisme coordenades */
#include <stdio.h>


int main (void)
{
    int   dx,dy,p,end;
    float x1,x2,y1,y2,x,y;

    scanf("%f", &x1);
    scanf("%f", &y1);
    scanf("%f", &x2);
    scanf("%f", &y2);

    if (x1>=x2)
        dx = (int)(x1-x2);
    else
        dx = (int)(x2-x1);

    if (y1>=y2)
        dy = (int)(y1-y2);
    else
        dy = (int)(y2-y1);

    p = 2 * dy - dx;
```

```
    if (x1>x2){
        x = x2;
        y = y2;
        end = (int)(x1);
    }else{
        x = x1;
        y = y1;
        end = (int)(x2);
    }

    printf("%.2f ", x);
    printf("%.2f ", y);

    while (x<(float)(end)){
          x = x + 1.0;
          if (p < 0)
             p = p + 2 * dy;
          else{
             y = y + 1.0;
             p = p + 2 * (dy-dx);
          }
          printf("%.2f ", x);
          printf("%.2f ", y);
    }
    return 0;
}
```