Índex

1 Problemes de FST/disseny descendent

1.1 Planetes

Es disposa de la informació corresponent a un conjunt de planetes d'una determinada galàxia. Cada planeta ve representat per la seva posició, velocitat i massa. Ens demanen que dissenyem un algorisme que simuli el moviment d'aquest conjunt de planetes. La simulació es farà durant NI intervals de temps de durada DT (NI i DT seran constants de l'algorisme). Per determinar les noves posicions i velocitats de cada planeta, caldrà aplicar les expressions clàssiques del moviment rectilini uniformement accelerat:

$$\boldsymbol{p} = \boldsymbol{p_0} + \boldsymbol{v_0}\, t + 0.5\boldsymbol{a}\, t^2$$

$$\boldsymbol{v} = \boldsymbol{v_0} + \boldsymbol{a}\, t$$

L'acceleració de cada planeta serà la resultant de l'aplicació del conjunt de forces que s'estableix entre tots els planetes, que segueixen la llei de gravitació clàssica:

$$G\frac{m_1\, m_2}{d^2},\ G = 6.67\, 10^{-11}\, N\, m^2\, kg^{-2}$$

Disposem de dos procediments que ens permeten carregar a memòria i visualitzar la llista de planetes.

acció $CarregaLlistaPlanetes$(**sor** $galaxia : tLlistaPlanetes$)
acció $VisualitzaLlistaPlanetes$(**ent** $galaxia : tLlistaPlanetes$)

El tipus per representar aquesta llista ens és conegut, és una taula de mida variable:

tipus
  $tTauPlanetes =$ **taula** $[1..MAXPLA]$ **de** $tPlaneta$
  $tLListaPlanetes =$ **tupla**
    n : **enter**
    $p : tTauPlanetes$
  ftupla
ftipus

NOTA: Cal fer abstracció del tipus per representar un planeta en un primer nivell d'anàlisi descendent i del tipus $Vector3D$ que permetrà representar vectors força, acceleració, velocitat i posició a l'espai, en nivells posteriors.

1.2 ReconeiImatges

Construïu un algoritme que permeti reconèixer una subimatge d'una imatge donada. Aquests algoritmes sovint s'anomenen algoritmes de *pattern matching* i tenen molta importància en el camp de la visió per computador. Les imatges per tractar estan formades per píxels de color blanc o negre. És a dir, la imatge o fotografia es representa com una malla quadrada de punts, i s'analitza el color de cada punt assignant-li blanc o negre segons la seva intensitat. A l'algoritme, les imatges li arriben pel canal estàndard d'entrada codificades de la següent manera:

```
7 5
.....
.xxxx
..x..
..x..
..x..
xxxx.
..x..
```

És a dir, nombre de files, nombre de columnes i imatge, on la imatge està formada per caràcters cadascun dels quals representa un píxel. La codificació és '.' per indicar el color blanc i 'x' per indicar el color negre. Les imatges com a màxim poden tenir 100 files i 200 columnes.

L'algoritme rep com a entrada dues imatges i cal que indiqui en quin lloc de la primera fa *matching* la segona, en cas que sigui així. Se sobrentén que una imatge fa *matching* amb una altra si els píxels de la primera i de la segona coincideixen un a un. En el cas que la segona imatge faciu matching en més d'una posició de la primera, l'algoritme indica només un dels llocs on fa el matching.

Per exemple, la imatge

```
3 3
xxx
.x.
.x
```

fa *matching* amb l'anterior a la posició (2,2).

1.3 DiferenciaImatges, exercici 1 de l'examen parcial del 1/04/2000

Considerem que una imatge en blanc i negre ve representada per les coordenades dels seus pixels negres. Es disposa de dos fitxers seqüencials de text (FST) anomenats *Imatge1* i *Imatge2* respectivament que contenen dues imatges representades en la forma indicada i es vol obtenir la imatge diferència en un tercer fitxer anomenat *Imatge3*.

Es pot suposar que:

- les coordenades dels pixels són dos enters
- les imatges estan ordenades primer per la coordenada y i després per la x.
- les seqüències acaben amb el pixel sentinella $(-1, -1)$

Es demana:

1. L'especificació de l'algorisme que dugui a terme l'esmentada diferència.
2. El disseny de l'algorisme justificant les seqüències que hi intervenen i els esquemes utilitzats.

Les imatges de la figura 1 es corresponen amb els següents fitxers:

Imatge1 = (2,2),(3,2),(4,2),(2,3),(4,3),(2,4),(3,4),(4,4),(-1,-1)
Imatge2 = (4,1),(4,2),(4,3),(4,4),(-1,-1)

Imatge3 = (2,2),(3,2),(2,3),(2,4),(3,4),(-1-1)

(Als fitxers no hi ha ni parèntesis ni comes. En aquest exemple s'hi han posat per claretat.)

Figura 1: a) Imatge1 b) Imatge2 c) Imatge3 = Imatge1 - Imatge2

1.4 compteCorr, exercici 1 de l'examen parcial del 1/04/2002

La caixa esmentada ens demana que dissenyem un subprograma que donat un compte corrent i l'interès diari (en tant per cent) que s'aplica, obtingui els interessos generats fins al moment.

Un compte corrent està format per una seqüència de moviments amb la corresponent data i moviment. Aquest pot ser un ingrés (valor positiu) o un càrrec (valor negatiu).

Els moviments estan ordenats per data i n'hi poden haver diversos amb la mateixa data. Es considera el saldo diari com el que s'obté després d'haver aplicat el darrer moviment del dia.

Les dates no tenen perquè ser consecutives, és a dir, si un dia no hi ha moviment, la data d'aquest dia no apareix i, per tant, s'entén que hi pot haver períodes de n dies consecutius en què no hi ha variació de saldo.

Per calcular l'interès de cada dia només cal aplicar l'interès diari al saldo diari. Podem suposar que el saldo diari és sempre positiu.

L'equip informàtic de la caixa ens proporciona els següents subprogrames que duen a terme operacions sobre el tipus tCC que representa un compte corrent i sobre el tipus $tData$:

{Prec: cc és un compte corrent no buit}
acció $InicialitzaCC$(**entsor** $cc : tCC$, **sor** $dataC : tData$, **sor** $saldoInic$: **real**)
{Post: $dataC$ és la data de creació del compte corrent i
$saldoInic$ és l'import inicial en Euros amb què es va obrir}
{Prec: cc és un compte corrent no buit}
acció $ObtMovCC$(**entsor** $cc : tCC$, **sor** $data : tData$, **sor** mov : **real**, **sor** $darrerM$: **booleà**)
{Post: $data$ és la data del moviment, mov és l'import del moviment (positiu o negatiu) i
$darrerM$ indica si aquest és el moviment sentinella}
{Prec: $dataA$ és anterior o igual a $dataB$}
funció $nDies$(**ent** $dataA, dataB : tData$)
{Post: $nDies(dataA, dataB) \geq 0$, és el nombre de dies transcorreguts entre $dataA$ i $dataB$}

Notes: L'acció *InicialitzaCC* obre un compte corrent, n'extreu la data d'obertura i el saldo inicial i se situa en el primer moviment. L'acció *ObtMovCC* obté el moviment en curs i situa el compte corrent al següent moviment. La funció *nDies* retorna 0 si les dates són iguals.

L'especificació d'aquest subprograma és opcional. Pel disseny, cal justificar les seqüències i esquemes usats.

1.5 Fitxers1, exercici 1 de l'examen parcial del 3/10/2003

Dissenyeu una funció que comprova si un enter x pertany a la seqüència d'enters emmagatzemada en el fitxer amb el nom que conté el paràmetre n, i que està especificada com segueix:

tipus
$Cadena$ = **taula** [0..30] **de caràcter**
ftipus
{Pre: el fitxer de nom n existeix i conté una seqüència d'enters que acaba en zero}
funció $pertanyFST$(**ent** x : **enter**, **ent** $n : Cadena$) **retorna booleà**
{Post: $pertanyFST(x, n)$ és cert si i només si x està al fitxer amb el nom contingut en n}

1.6 Substitucio1, exercici 1 de l'examen parcial del 3/03/2004

Es té un fitxer anomenat "discurs" estructurat com una seqüència de paraules (o mots) separades només per espais i finalitzada per una paraula sentinella. Dissenyeu una acció tal que donades dues paraules $mot1$ i $mot2$, generi un fitxer anomenat "discursCorregit" que serà igual que el fitxer "discurs" excepte aquelles paraules que són idèntiques a $mot1$, les quals en el fitxer FST "discursCorregit" apareixeran substituides pel $mot2$.

Podeu comptar amb un tipus, $tMot$, que representa una paraula amb els següents subprogrames definits:

acció $llegirMot$(**entsor** $f : FST$, **sor** $m : tMot$)
{**pre:** A través de f s'accedeix al fitxer seqüencial de text T, i T és una seqüència de mots $M_1 M_2 \dots M_f$. $mode(f) = lectura$ i $FdFST(f) = fals$ i $pd(f) = M_i \cdot \gamma$, $1 \leq i \leq n$}
{**post:** $mode(f) = lectura$ i m representa a M_i i $pd(f) = \gamma$}

acció $escriureMot$(**entsor** $f : FST$, **ent** $m : tMot$)
{**pre:** $mode(f) = escriptura$ i $pe(f) = \alpha$ i $m = M$}
{**post:** $mode(f) = escriptura$ i $pe(f) = \alpha \cdot M \cdot \sqcup$ on $\sqcup$és el caràcter espai}

funció $motsIguals$(**ent** $m1 : tMot$, **ent** $m2 : tMot$) **retorna booleà**
{**pre:** $m1 = M1$ i $m2 = M2$}
{**post:** $motsIguals(m1, m2)$ és cert si i només si $M1$ i $M2$ representen el mateix mot.}

funció $motSentinella$(**ent** $m : tMot$) **retorna booleà**
{**pre:** $m = M$}
{**post:** $motSentinella(m)$ és cert si i només si M és el mot sentinella.}

1.7 ControlHorari, exercici 1 de l'examen parcial del 2/10/2004

L'empresa CONTROL S.A. vol conèixer quin dels seus empleats ha treballat més hores al llarg d'un periode de temps. Per això, ens demana un algorisme que a partir de les dades introduïdes per un operador ens mostri pel canal estàndar de sortida el nom de l'empleat amb més hores treballades, i quantes hores de més ha fet respecte de la mitjana d'hores treballades per tots els empleats.

Les dades venen pel CEE en aquest ordre posant per cada empleat, el seu nom (com a molt 30 caràcters), seguit d'una seqüència de dades provinents d'un rellotge de control de presència que indica les seves entrades i sortides a l'empresa. La seqüència acabarà amb un empleat sentinella.

Dissenyeu l'algorisme demanat utilitzant les següents accions i funcions donades.
Notes: En el cas poc probable de coincidència d'hores, l'empleat afortunat serà el primer que trobem en la seqüència. Com a mínim hi haurà un empleat vàlid en la seqüència.

acció $llegirEmpleat$(**sor** $e : tEmpleat$)
{**Pre:** $s \equiv e_1 \dots e_n e_f$, $pd(s) = e_i \cdot \beta$, i $\beta = e_{i+1} \dots e_f$, $1 \leq i \leq f$ }
{**Post:** e representa e_i, i $en\_curs(S) = e_i$ i $pd(S) = \beta$}

funció $empleatSentinella$(**ent** $e : tEmpleat$) **retorna booleà**
{**Pre:** $e = E$}
{**Post:** empleatSentinella(e) és cert si E és l'empleat sentinella. Fals en cas contrari.}

acció *escriureEmpleat*(**ent** $e : tEmpleat$)
{**Pre:** $e = E$}
{**Post:** Pel CSS surt N i H, on N és el nom d'E i H són les hores treballades d'E.}
funció *hores*(**ent** $e : tEmpleat$) **retorna real**
{**Pre:** $e = E$}
{**Post:** hores(e) és el nombre d'hores treballades per l'empleat E}

1.8 RPM, exercici 1 de l'examen parcial del 1/04/2005

Per tal de programar un xip d'injecció d'un cotxe ens demanen que dissenyem un algorisme. Com a entrada tindrà les dades d'un fitxer anomenat "sensors.dat" que conté una seqüència de tripletes de reals que indiquen les revolucions per minut a les que està funcionant el motor, la temperatura del motor, i la velocitat del cotxe.

L'algorisme calcularà el cabal de combustible que ha de disparar l'injector en funció de les revolucions per minut. Per fer-ho disposareu d'una funció *cabalAdient* que a partir de les revolucions per minut us dirà quin cabal cal aplicar, i un altre, *cabalReduit*, que segons les revolucions per minut, temperatura del motor i la velocitat del cotxe indica quin cabal aplicar. La primera funció s'aplica sempre que la temperatura del motor és menor que 100 $^\circ C$ i la velocitat del cotxe és menor que 200 Km/h. En cas contrari s'aplicarà la segona funció. Tots els cabals calculats s'escriuran a un fitxer de sortida anomenat "cabals.dat".

Dissenyeu l'algorisme demanat considerant que les següents accions i funcions ja estan dissenyades.

acció *llegirSensors*(**entsor** $f : FST$, **sor** rpm : **real**, **sor** $temp$: **real**, **sor** $veloc$: **real**)
{**Pre:** f és un FST de (r,t,v), on r és el nombre de revolucions per minut, t la temperatura, i v la velocitat, i $mode(f) = lectura$ i $pe(f) = \alpha$ i $en\_curs(f) = \beta$ i $pd(f) = \gamma\delta$ i $pd(f) \neq \emptyset$ i $\beta = R \cdot T \cdot V$}
{**Post:** $mode(f) = lectura$ i $pe(f) = \alpha \cdot \beta \wedge en\_curs(f) = \gamma \wedge pd(f) = \delta$ i rpm representa R en revolucions per minut, $temp$ representa la temperatura en $^\circ C$ i V la velocitat en Km/h}
funció *sensorSentinella*(**ent** rpm : **real**, **ent** t : **real**, **ent** v : **real**) **retorna booleà**
{**Pre: cert**}
{**Post:** sensorSentinella(r,t,v) és cert si r, t, v són una tripleta sentinella. Si no ho és, retorna fals}
funció *cabalAdient*(**ent** rpm : **real**) **retorna real**
{**Pre:** $rpm = RPM$ i RPM està en revolucions per minut}
{**Post:** $cabalAdient(r)$ és el cabal adient per el valor RPM }
funció *cabalReduit*(**ent** rpm : **real**, **ent** t : **real**, **ent** v : **real**) **retorna real**
{**Pre:** $rpm = RPM$ i RPM està en revolucions per minut i $t = T$ i T està en $^\circ C$, i $v = V$ està en Km/h}
{**Post:** $cabalReduit(r, t, v)$ és el cabal que cal aplicar pels valors RPM, T i V }

1.9 ControlEntSor, exercici 1 de l'examen parcial del 2/10/2005

Una botiga vol conèixer quina és l'hora de màxima afluència de clients al llarg d'un dia. Per a fer-ho col·loca dos sensors, un a cada porta del local. Cada sensor compta quanta gent entra i surt cada minut. Cada sensor guarda els resultats en un fitxer, `sensor1.txt` i `sensor2.txt` respectivament. Les dades que guarden els sensors tenen següent format:

$$H_1\ M_1\ E_1\ S_1\ H_2\ M_2\ E_2\ S_2\ \ \ldots\ \ -1\ -1\ E_s\ S_s$$

on H_i és un enter que representa l'hora, M_i és un enter que representa el minut, l'enter E_i és el nombre de persones que han entrat en el moment $H_i\ M_i$ i l'enter S_i és el nombre de persones que han sortit en aquest minut. Les dades s'enregistren ordenades segons l'hora temporalment en els fitxers. Els sensors només enregistren una dada al seu fitxer quan detecta

una entrada o sortida de clients (és a dir, no escriu un minut si no ha entrat ni ha sortit ningú). El primer registre $H_1\ M_1\ E_1\ S_1$ del fitxer correspon a la primera hora del dia en la qual entren clients a la botiga. El darrer registre és una marca de final que es caracteritza perquè tant l'hora com el minut valen -1.

Es disposa de:

La definició del tipus *Registre*, que representa cada un dels registres que hi ha en els fitxers donats, és a dir, representa la informació de cada minut, i de la definició del tipus *Hora* que representa un moment (hora i minuts). A més, ens donen operacions que treballen amb objectes dels tipus definits.

tipus
Registre
Hora
ftipus

acció $LlegirRegistreFST$(**entsor** $f : FST$, **sor** $r : Registre$)
{**Pre:** f es una referència a un fitxer de registres (H_i, M_i, E_i, S_i), i $mode(f) = lectura \ \wedge \ pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \cdot \delta \ \wedge \ pd(f) \neq \emptyset$}
{**Post:** $mode(f) = lectura \quad \wedge \quad pe(f) = \alpha.\beta \quad \wedge \quad en\_curs(f) = \gamma \quad \wedge \quad pd(f) = \delta \quad \wedge$ r representa el registre $\gamma \quad (H_i, M_i, E_i, S_i)$}

funció $RegistreSentinella$(**ent** $r : Registre$) **retorna booleà**
{**Pre:** }
{**Post:** RegistreSentinella(r) és cert si tant l'hora com el minut representat en r són -1. És fals en cas contrari}

funció $MovClients$(**ent** $r : Registre$) **retorna enter**
{**Pre:** r representa (H, M, E, S)}
{**Post:** $MovClients(r) = E - S$}

funció $HoraRegistre$(**ent** $r : Registre$) **retorna** $Hora$
{**Pre:** r representa (H, M, E, S)}
{**Post:** $Hora(r)$ representa (H, M)}

acció $EscriureHora$(**ent** $h : Hora$)
{**Pre:** h representa $(H, M) \wedge CSS = \gamma$}
{**Post:** $CSS = \gamma\{H, M\}$}

funció $EsAbans$(**ent** $h1, h2 : Hora$) **retorna booleà**
{**Pre:** }
{**Post:** $EsAbans(h1, h2)$ és cert si l'hora representada per $h1$ és anterior a la representada per $h2$. En cas contrari és fals.}

funció $Alhora$(**ent** $h1, h2 : Hora$) **retorna booleà**
{**Pre:** }
{**Post:** Alhora(h1, h2) és cert si h1 i h2 representen la mateixa hora. En cas contrari és fals.}

Es demana:

- *(5 punts)* Dissenyeu un algorisme que a partir de les dades dels dos FST escrigui pel CES l'hora (hora i minut) de màxima afluència, és a dir, el moment en que el nombre de clients que hi ha a la botiga és màxim, així com el nombre de clients que hi ha hagut en aquest moment. Suposeu que cap de les dues seqüències dels dos fitxers és buïda.
- *(1 punt)* Definir el tipus Registre de manera raonada.
- *(1 punt)* Definir el tipus Hora i dissenyeu els subprogrames $EscriureHora$(**ent** $h1 : Hora$) i $EsAbans$(**ent** $h1, h2 : Hora$)

1.10 Cromos, exercici 1 de l'examen parcial del 0/04/2006

Els menuts de la casa us demanen un algorisme que els ajudi a intercanviar cromos amb els amics. L'algorisme rep les dades en dos fitxers seqüencials de text anomenats `"jo"` i `"tu"`. Cada un dels fitxers conté una seqüència de cromos ordenats estrictament creixent i acabada amb un cromo sentinella. Els fitxers `"jo"` i `"tu"` contenen els cromos que es volen intercanviar dues persones. L'algorisme ha de calcular els fitxers `"jonoelstinc"` i `"tunoelstens"`. El fitxer `"jonoelstinc"` contindrà els cromos del fitxer `"tu"` que no són al fitxer `"jo"` mentre que el fitxer `"tunoelstens"` contindrà els cromos del fitxer `"jo"` que no són al fitxer `"tu"`.

Per tal de dissenyar l'algorisme disposeu del tipus *Cromo* i dels subprogrames següents:

acció $LlegirCromoFST$(**entsor** $f : FST$, **sor** $c : Cromo$)
{**Pre:** f obert per lectura }
{**Post:** c és el següent element d'f }

acció $EscriureCromoFST$(**entsor** $f : FST$, **ent** $c : Cromo$)
{**Pre:** f obert per escriptura }
{**Post:** c s'ha afegit al final d'f }

funció $CromoSentinella$(**ent** $c : Cromo$) **retorna booleà**
{**Pre:** cert }
{**Post:** c és el cromo sentinella }

funció $IgualCromo$(**ent** $c1, c2 : Cromo$) **retorna booleà**
{**Pre:** cert }
{**Post:** $c1$ és igual a $c2$ }

funció $MenorCromo$(**ent** $c1, c2 : Cromo$) **retorna booleà**
{**Pre:** cert }
{**Post:** $c1$ és menor que $c2$ }

Els cromos de la col·lecció d'aquest problema són de futbolistes. A cada cromo hi consta el nom d'un futbolista i l'equip al qual pertany. Els cromos estan ordenats primer per equip i dins de cada equip, pel nom del futbolista. Per exemple, el cromo d'en Pere del Balaguer és més petit que el d'en Jaume de l'Hospitalet, però el d'en Pere del Balaguer és més gran que el d'en Jordi, també del Balaguer. Dissenyeu la funció *MenorCromo* especificada en el problema 1 suposant que disposeu del tipus *Paraula* i dels subprogrames següents:

funció $EquipCromo$(**ent** $c : Cromo$) **retorna** $Paraula$
{**Pre:** cert }
{**Post:** retorna l'equip al qual pertany el futbolista del cromo c}

funció $NomCromo$(**ent** $c : Cromo$) **retorna** $Paraula$
{**Pre:** cert }
{**Post:** retorna el nom del futbolista del cromo c}

funció $IgualParaula$(**ent** $p1, p2 : Paraula$) **retorna booleà**
{**Pre:** cert }
{**Post:** $p1$ és igual a $p2$ }

funció $MenorParaula$(**ent** $p1, p2 : Paraula$) **retorna booleà**
{**Pre:** cert }
{**Post:** $p1$ és menor que $p2$ }

1.11 Matricula, exercici 1 de l'examen parcial del 3/10/2006

Es tenen dos fitxers de text: "`notesq20506.txt`" i "`matq10607.txt`". El primer fitxer, "`notesq20506.txt`", conté informació sobre els estudiants que estaven matriculats a l'assignatura d'Informàtica durant el quadrimestre de primavera del curs 05/06. Cada registre d'aquest fitxer és del tipus $tNotaEst$, que conté les dades personals de l'estudiant (nom, DNI,

...), el nombre de convocatòria de matricula de la primavera 05/06 i la nota que va treure de l'assignatura.

El segon fitxer conté les dades personals (nom, DNI,...) dels estudiants que es poden matricular d'Informàtica el quadrimestre de tardor del curs 06/07. Cada registre és del tipus $tNouEstudiant$.

Es demana construir un nou fitxer "`matpreuq10607.txt`" que contingui, per tots els estudiants que es poden matricular d'Informàtica el quadrimestre de tardor del curs 06/07, és a dir pels estudiants de "`matq10607.txt`", a més de les dades personals, la convocatòria que representarà aquest quadrimestre de tardor i el preu de la matrícula d'aquesta assignatura.

Per fer-ho hem de tenir en compte:

- Els dos fitxers estan ordenats segons els DNI dels estudiants en ordre creixent.
- Els dos fitxers acaben amb un sentinella.
- El preu de la matricula és igual a PREU euros si és la primera convocatòria. PREU és una constant que podeu suposar definida. Aquest preu s'incrementa un 30% si es repeteix, però encara no s'ha arribat a la 5ena convocatòria. Si ja s'ha repetit abans quatre cops o més, l'increment del preu és d'un 50%.

Podeu utilitzar els següents subprogrames:

{**Pre:** f obert per lectura }
acció $llegirNotaEstFST$(**entsor** $f : FST$, **sor** $est : tNotaEst$)
{**Post:** est és el següent element $tNotaEst$ d'f }

{**Pre:** f obert per lectura }
acció $llegirNouEstFST$(**entsor** $f : FST$, **sor** $est : tNouEstudiant$)
{**Post:** est és el següent element $tNouEstudiant$ d'f }

{**Pre:** f obert per escriptura }
acció $escriureNouEstFST$(**entsor** $f : FST$, **ent** $est : tNouEstudiant$)
{**Post:** les dades de l'estudiant, est s'ha afegit al final d'f }

{**Pre:** $est = EST$ }
funció $DNIEst$(**ent** $est : tNotaEst$) **retorna enter**
{**Post:** $DNIEst(est)$ és el DNI de l'estudiant d'EST}

{**Pre:** $est = EST$ }
funció $DNINouEst$(**ent** $est : tNouEstudiant$) **retorna enter**
{**Post:** $DNINouEst(est)$ és el DNI de l'estudiant d'EST}

{**Pre:** $est = EST$ }
funció $NotaNEst$(**ent** $est : tNotaEst$) **retorna real**
{**Post:** $NotaNEst(est)$ és la nota de l'estudiant d'EST}

{**Pre:** $est = EST$ }
funció $ConvocNEst$(**ent** $est : tNotaEst$) **retorna enter**
{**Post:** $ConvocNEst(est)$ és el nombre de convocatòria de l'estudiant d'EST}

{**Pre:** $est = EST$ }
funció $SentNEst$(**ent** $est : tNotaEst$) **retorna booleà**
{**Post:** $SentNEst(est)$ és cert si EST és el sentinella}

{**Pre:** $est = EST$ }
funció $SentNouEst$(**ent** $est : tNouEstudiant$) **retorna booleà**
{**Post:** $SentNouEst(est)$ és cert si EST és el sentinella}

Es vol transformar l'algorisme anterior en un subprograma genèric que admet com a dades els noms de fitxer amb els que ha de treballar.

a) Escriviu la capçalera del subprograma.
b) Escriviu un tros d'algorisme que cridi al subprograma del punt a). En aquesta crida, el fitxer amb registres de tipus $tNotaEst$ s'anomena "`notesmecq20506.txt`", el fitxer amb registres de tipus $tNouEstudiant$ s'anomena "`matmecq10607.txt`" i el fitxer resultant s'anomena "`preumec0607.txt`". Aquest tros d'algorisme ha d'incloure la declaració i la inicialització dels paràmetres actuals de la crida del subprograma.

1.12 Circumscrites

En un FST anomenat "Circumferències" hi ha un parell de circumferències a cada registre. Una circumferència ve donada per tres reals (les coordenades x i y del centre i el *radi*). Al final del fitxer hi ha un registre sentinella amb dues circumferències de radi 0. Ens asseguren que les dues circumferències de cada parell són disjuntes. Dissenyeu un algoritme que, a partir del fitxer Circumferències, en creï un altre anomenat "Circumscrites" on hi hagi el centre i radi de la circumferència que circumscriu a cada parell i la diferència d'àrees entre la circumferència circumscriptora i les dues a les que cirscumscriu.

Es disposa del següent subprograma:

{Prec: *cert*}
funció $CircumfCircums$(**ent** $c1, c2 : Circumferencia$)
retorna $Circumferencia$
{Post: $CircumfCircums(c1, c2) = c3 \ \wedge$ $c3$ és la circumferència
que circumscriu a $c1$ i $c2$}

Es recomana treballar amb operacions del tipus $Circumferencia$ i, a continuació, definir aquest tipus i implementar-ne les operacions.

1.13 Obstacles, exercici 2 de l'examen parcial del 1/04/2001

En una planta d'una empresa hi ha una sèrie d'obstacles (columnes, mobles, etc.) i s'està dissenyant un robot que s'haurà de moure per aquesta planta seguint un determinat camí. Ens demanen que dissenyem un subprograma que indiqui quins d'aquests obstacles molesten, és a dir, són interceptats pel camí que ha de seguir el robot.

Els obstacles es representen pel tipus definit *Obstacle* i es troben en un fitxer anomenat "Planta". Al final d'aquest fitxer hi ha un obstacle sentinella. Els obstacles que molesten s'hauran d'escriure en un altre fitxer anomenat "Destorb".

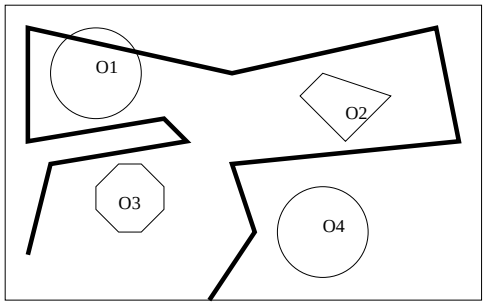


Figura 2: Planta amb 4 obstacles i un cami amb 12 punts.

Les operacions que podem fer amb els obstacles es representen amb els següents subprogrames.

{Prec: $mode(f) = lectura \ \wedge \ pe(f) = \alpha$
$\wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = o.\gamma \ \wedge \ pd(f) \neq \emptyset$}
acció $LlegirObstFST$(**entsor** $f : FST$, **sor** $o : Obstacle$)
{Post: $mode(f) = lectura \ \wedge \ pe(f) = \alpha.\beta \ \wedge \ o = en\_curs(f) \ \wedge \ pd(f) = \gamma$}
{Prec: $mode(f) = escriptura \ \wedge \ pe(f) = \alpha$}
acció $EscriureObstFST$(**entsor** $f : FST$, **ent** $o : Obstacle$)
{Post: $mode(f) = escriptura \ \wedge \ pe(f) = \alpha.o$}
{Prec: $cert$}
funció $ObstSentinella$(**ent** $o : Obstacle$) **retorna booleà**
{Post: $ObstSentinella(o) \Longleftrightarrow o$ és l'obstacle sentinella}
{Prec: $cert$}
funció $ObstMolesta$(**ent** $o : Obstacle$, **ent** $p, q : Punt$) **retorna booleà**
{Post: $ObstMolesta(o) \Longleftrightarrow o$ és interceptat pel segment de recta $\overline{pq}$}

El camí és una poligonal, és a dir, una seqüència de segments de línia recta que ve representada pel tipus *Cami*. Vegeu la figura adjunta.

Es demana:

1. Especifiqueu el subprograma que a partir d'un camí escrigui al fitxer "Destorb" aquells obstacles que es troben al fitxer "Planta" i que molesten.

2. Implementeu aquest subprograma fent ús de les operacions del tipus Obstacle.

   - Caracteritzeu la seqüència a tractar i indiqueu l'esquema que fareu servir.
   - En la implementació s'han d'usar les operacions del tipus Obstacle. També s'ha d'especificar i usar un altre subprograma que a partir d'un obstacle i el camí indiqui si l'obstacle molesta. Aquest subprograma s'implementarà després. Encara no s'ha de definir el tipus *Cami*.

3. Implementeu el subprograma, especificat a l'apartat anterior, que a partir d'un obstacle i un camí indica si l'obstacle molesta. Podeu fer ús del tipus *Cami* definit com segueix:

```
tipus
  TauPunts = taula [1..100] de Punt
  Cami = tupla
           t : TauPunts
           n : enter
         ftupla
ftipus
```

   Caracteritzeu la seqüència a tractar i indiqueu l'esquema que fareu servir.

1.14 Vacances, exercici 2 de l'examen parcial del 3/10/2002

Una agència de viatges disposa de dos fitxers "NovaYork.txt" i "Bali.txt" amb les dades dels clients que han viatjat a aquestes dues destinacions turístiques. El format dels dos fitxers és el mateix: cada línia del fitxer conté el DNI del client (enter), nom (de tipus *Paraula*) i

l'adreça postal (de tipus *Paraula*). Els dos fitxers estan ordenats pel DNI i acaben amb un client fictici de DNI -1.

Dissenyeu un algoritme que a partir de les dades indicades obtingui dos altres fitxers, "nomesBali.txt" i "nomesNY.txt"; el primer ha de contenir la llista de clients que només han elegit Bali com a destinació turística (però no Nova York), i el segon la llista de clients que només han viatjat a Nova York (però no a Bali). El format d'aquests altres fitxers ha de ser el mateix que el dels dos originals. Cal que indiqueu l'esquema algorísmic aplicat. Pel disseny heu d'usar les operacions del tipus *Paraula*, especificades tot seguit i l'especificació del tipus FST que hi ha a l'altre full.

{Prec: $mode(f) = lectura \ \wedge \ pe(f) = \alpha \ \wedge \ Paraula\_en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma\delta$}
acció *LlegirParaulaFST*(**entsor** $f : FST$, **sor** $p : Paraula$)
{Post: $mode(f) = lectura \ \wedge \ pe(f) = \alpha.\beta \ \wedge \ p = Paraula\_en\_curs(f) = \gamma \ \wedge \ pd(f) = \delta$}
{Prec: $mode(f) = escriptura \ \wedge \ pe(f) = \alpha$}
acció *EscriureParaulaFST*(**entsor** $f : FST$, **ent** $p : Paraula$)
{Post: $mode(f) = escriptura \ \wedge \ pe(f) = \alpha.p$}

1.15 FussioFitxers, exercici 2 de l'examen parcial del 3/10/2003

Es disposa de dos fitxers seqüencials de text "Sous" i "ObjectesLuxosos". El primer dels fitxers està format per registres que contenen informació sobre el sou d'individus. El segon fitxer està format per registres d'objectes luxosos contenint informació sobre el seu cost i el seu propietari.

Ambdós tipus de registres contenen el DNI. Els dos fitxers estan ordenats per DNI de forma creixent i acaben amb un sentinella donat pel DNI igual a 0. Es demana fer un algorisme que construeixi un fitxer amb el DNI de tots els individus amb un o més objectes luxosos fora de les seves possibilitats. Es disposen de dos tipus definits: *tSou* per recollir la informació d'un registre del fitxer "Sous", i *tObjLuxos* per recollir la informació d'un registre del fitxer "ObjectesLuxosos". A més podeu utilitzar els subprogrames següents que ja estan desenvolupats:

acció $LlegirSouFST$(**entsor** $f : FST$, **sor** $s : tSou$)
{ **Pre:** f es un FST lligat a un fitxer de registres de sous, i $mode(f) = lectura \ \wedge \ pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \cdot \delta \ \wedge \ pd(f) \neq \emptyset$ }
{ **Post:** $mode(f) = lectura \ \wedge \ pe(f) = \alpha.\beta \ \wedge \ en\_curs(f) = \gamma \ \wedge \ pd(f) = \delta \ \wedge \ s$ representa γ }
funció $SentinellaSou$(**ent** $s : tSou$) **retorna booleà**
{ **Pre:** cert }
{ **Post:** $SentinellaSou(s)$ és cert si s és el sou sentinella. Si no ho és, retorna fals }
funció $DNISou$(**ent** $s : tSou$) **retorna enter**
{ **Pre:** cert }
{ **Post:** $DNISou(s)$ retorna el DNI del registre de sou S }
acció $LlegirObjLuxosFST$(**entsor** $f : FST$, **sor** $o : tObjLuxos$)
{ **Pre:** f es un FST lligat a un fitxer de registres d'objectes luxosos, i $mode(f) = lectura \ \wedge \ pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \cdot \delta \ \wedge \ pd(f) \neq \emptyset$ }
{ **Post:** $mode(f) = lectura \ \wedge \ pe(f) = \alpha \cdot \beta \ \wedge \ en\_curs(f) = \gamma \ \wedge \ pd(f) = \delta \ \wedge \ o$ representa γ }
funció $SentinellaObjLuxos$(**ent** $o : tObjLuxos$) **retorna booleà**
{ **Pre:** cert }
{ **Post:** $SentinellaObjLuxos(o)$ és cert si o és l'objecte luxós sentinella. Si no ho és, retorna fals }
funció $DNIObjLuxos$(**ent** $o : tObjLuxos$) **retorna enter**
{ **Pre:** cert }
{ **Post:** $DNIObjLuxos(o)$ retorna el DNI de l'objecte luxós o }

funció $DinsDePossibilitats$(**ent** $o : tObjLuxos$, **ent** $s : tSou$) **retorna booleà**
{ **Pre:** cert }
{ **Post:** $DinsDePossibilitats(o, s)$ retorna cert si la pertinença de l'objecte luxós està dins de les possibilitats del sou del propietari. En cas contrari, retorna fals }

Notes: Un mateix individu pot tenir més d'un objecte luxós o bé no tenir-ne cap.

Cada individu té un únic sou.

El fitxer de sortida ha de contenir registres de DNI únics; és a dir, sense repetició.

1.16 Substitucio2, exercici 2 de l'examen parcial del 3/03/2004

Es disposa d'un tipus $tMot$ el qual representa una paraula formada per una seqüència de paraules. Tenim les següents operacions bàsiques:

funció $creaMot()$ **retorna** $tMot$
{**pre: cert**}
{**post:** creaMot() representa el mot buit}

acció $afegirCaracterMot$(**ent** c : **caràcter**, **entsor** $m : tMot$)
{**pre:** $M = c_1 \cdot c_2 \ldots \cdot c_n$ i $c = C$}
{**post:** $M = c_1 \cdot c_2 \ldots \cdot c_n \cdot C$}

acció $primerCaracterMot$(**entsor** $m : tMot$, **sor** c : **caràcter**)
{**pre:** M representa $c_1 \cdot c_2 \ldots \cdot c_n$ i M no és buit}
{**post:** $C = c_1$}

acció $seguentCaracterMot$(**entsor** $m : tMot$, **sor** c : **caràcter**)
{**pre:** $m = M$ i M representa $c_1 \cdot c_2 \ldots \cdot c_n$ i $pd(M)$ no és buida i $pd(M) = c_i \cdot \gamma$, $1 \leq i \leq n$}
{**post:** $C = c_i$ i $pd(M) = \gamma$}

funció $caracterSentinellaMot$(**ent** c : **caràcter**) **retorna booleà**
{**pre:** $c = C$}
{**post:** caracterSentinellaMot(c) és cert si i només si C és el caràcter sentinella d'un mot $tMot$}

Dissenyeu a partir d'aquests subprogrames una funció $motsIguals$ amb la següent especificació:

funció $motsIguals$(**ent** $m1 : tMot$, **ent** $m2 : tMot$) **retorna booleà**
{**pre:** m1 = M1 i m2 = M2 $c = C$}
{**post:** motsIguals(m1, m2) és cert si i només si M1 i M2 representen el mateix mot}

1.17 FactModels, exercici 2 de l'examen parcial del 2/10/2004

Un concessionari de cotxes de diverses marques i models vol conèixer la seva facturació anual. Per fer-ho disposa de dos fitxers. El primer, anomenat "vendes.dat", conté una seqüència de parelles on cadascuna conté el nom del model d'un cotxe, i el nombre de cotxes venuts d'aquest model. El segon fitxer, "llistaDePreus.dat", conté també una seqüència de parelles on cadascuna conté el nom del model d'un cotxe, i el seu preu de venda al públic. Les parelles d'ambdós fitxers estan ordenades segons el nom de models de cotxe. Ambdós fitxers acaben amb una parella que té com a model un mot sentinella.

Es demana que dissenyeu un algorisme eficient que donats el fitxer "vendes.dat" i el fitxer "llistaDePreus.dat", escrigui la facturació total de les vendes efectuades. El fitxer de vendes només podrà contenir models de cotxe ja existents en el fitxer de preus.

Podeu comptar amb les accions:

acció $llegirVendaFST$(**entsor** $f : FST$, **sor** $m : tMot$, **sor** n : **enter**)
{**Pre:** f és un FST lligat a un fitxer de parelles de model cotxe i quantitat, i $mode(f) = lectura \ \wedge$ $pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \cdot \delta \ \wedge \ pd(f) \neq \emptyset$}
{**Post:** $mode(f) = lectura \ \wedge \ pe(f) = \alpha.\beta \ \wedge \ en\_curs(f) = \gamma \ \wedge \ pd(f) = \delta \ \wedge$ m i n representen el model de cotxe i la quantitat respectivament de la parella γ, i $n \geq 0$}

acció $llegirPreuFST$(**entsor** $f : FST$, **sor** $m : tMot$, **sor** p : **real**)
{**Pre:** f es un FST lligat a un fitxer de parelles de model cotxe i preu, i $mode(f) = lectura \ \wedge \ pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \cdot \delta \ \wedge \ pd(f) \neq \emptyset$}
{**Post:** $mode(f) = lectura \ \wedge \ pe(f) = \alpha.\beta \ \wedge \ en\_curs(f) = \gamma \ \wedge \ pd(f) = \delta \ \wedge$ m i p representen el model de cotxe i el preu respectivament de la parella γ, i $p \geq 0$}

funció $modelSentinella$(**ent** $m : tMot$) **retorna booleà**
{**Pre:** cert}
{**Post:** $modelSentinella(m)$ és cert si m és el model de cotxe sentinella. Si no ho és, retorna fals}

funció $motsIguals$(**ent** $m1 : tMot$, **ent** $m2 : tMot$) **retorna booleà**
{**Pre:** $m1 = M1$ i $m2 = M2$}
{**Post:** $motsIguals(m1, m2)$ és cert si i només si $M1$ i $M2$ representen el mateix mot.}

funció $mot1AbansMot2$(**ent** $m1 : tMot$, **ent** $m2 : tMot$) **retorna booleà**
{**Pre:** $m1 = M1$ i $m2 = M2$}
{**Post:** $mot1AbansMot2(m1, m2)$ és cert si i només si $M1$ va alfabèticament abans que $M2$ i $M1 \neq M2$.}

Per ex., si els fitxers, un cop hem tret el sentinella, són:

"vendes.dat"	"llistaDePreus.dat"
Focus 40	Astra 13000
Golf 50	Focus 10000
Mondeo 30	Galaxy 30000
	Golf 18500
	Laguna 8600
	Leon 15000
	Mondeo 12500
	Neon 18000
	Polo 9000
	Serena 12000
	Voyager 35000
	Z3 12000

L'algorisme escriurà la següent sortida: 1700000

1.18 ClassifPortals, exercici 2 de l'examen parcial del 1/04/2005

Una empresa de serveis d'internet té dos servidors de pàgines. Cada servidor genera automàticament un fitxer seqüencial de text que conté els noms de les pàgines que han estat visitades amb el seu nombre de visites. Les pàgines en cada fitxer estan ordenades de **major a menor** segons el nombre de visites que han rebut. A partir dels dos fitxers (de noms "pagines1.dat" i "pagines2.dat" respectivament) es vol generar un altre fitxer (de nom "les40millors.dat") que contingui **com a molt** les 40 primeres pàgines més visitades dels dos servidors.

En el cas que dues pàgines tinguin el mateix nombre de visites no importa com es classifiquen entre sí. Inclús si hi ha més de 40 nombres de visites iguals, s'agafaran només les primeres 40 que s'han llegit. Els dos servidors no tenen noms de pàgines iguals.

Podeu comptar amb les accions ($tMot$ ja està fet):

acció $llegirPaginaFST$(**entsor** $f : FST$, **sor** $nom : tMot$, **sor** $visites$: **enter**)
{**Pre:** f es un FST lligat a un fitxer de parelles de nom de pàgina i nombre de visites, i $mode(f) = lectura \ \wedge \ pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \cdot \delta \ \wedge \ pd(f) \neq \emptyset$}
{**Post:** $mode(f) = lectura \ \wedge \ pe(f) = \alpha.\beta \ \wedge \ en\_curs(f) = \gamma \ \wedge \ pd(f) = \delta \ \wedge$ nom i $visites$ representen el nom de la pàgina i el nombre de visites respectivament de la parella γ, i $visites \geq 0$}

funció $paginaSentinella$(**ent** $m : tMot$) **retorna booleà**
{**Pre:** cert}
{**Post:** $paginaSentinella(m)$ és cert si m és la pàgina sentinella. En cas contari $paginaSentinella(m)$ és fals}

acció $escriurePaginaFST$(**entsor** $f : FST$, **ent** $nom : tMot$, **ent** $visites$: **enter**)
{**Pre:** $mode(f) = escriptura$ i $pe(f) = \alpha$ i $nom = NOM$ i $visites = VISITES$}
{**Post:** $mode(f) = escriptura$ i $pe(f) = \alpha \cdot NOM \cdot \sqcup VISITES \cdot \sqcup$ on $\sqcup$és el caràcter espai}

Per exemple, si els continguts dels fitxers són:

"pagines1.dat"		"pagines2.dat"	
noticies.html	10000	jocsGratis.html	13000
cerca.html	5000	programes.html	10000
fotos.html	3000	compres.html	9000
		historia.html	500
		musica.html	500

L'algorisme escriurà el següent fitxer "les40millors.dat" :

jocsGratis.html	13000
noticies.html	10000
programes.html	10000
compres.html	9000
cerca.html	5000
fotos.html	3000
historia.html	500
musica.html	500

2 Problemes de complexitat

2.1 Complexitat, exercici 3 de l'examen parcial del 3/10/2003

En una planta d'una empresa hi ha una sèrie d'obstacles (columnes, mobles, etc.) i s'està dissenyant un robot que s'haurà de moure per aquesta planta seguint un determinat camí. Per això, s'ha dissenyat uns subprogrames que indiquen quins d'aquests obstacles molesten, és a dir, són interceptats pel camí que ha de seguir el robot. Els obstacles es representen pel tipus definit *Obstacle* i es troben en un fitxer anomenat "Planta". Al final d'aquest fitxer hi ha un obstacle sentinella. Els obstacles que molesten s'escriurà en un altre fitxer anomenat "Destorb". Els subprogrames son els següents:

{Prec: "`Planta`" $= o_1 o_2 \dots o_n o_s \ \wedge \ \forall i : 1 \leq i \leq n : o_i$ és un obstacle $\ \wedge$
o_s és l'obstacle sentinella $\ \wedge \ c$ és un camí}
{Post: "`Planta`" no es modifica $\ \wedge$ "`Destorb`" $= d_1 d_2 \dots d_p \ \wedge$
$\forall i : 1 \leq i \leq p : (\exists k : 1 \leq k \leq n : d_i = o_k \ \wedge \ d_i$ és interceptat per $c) \ \wedge$ "`Destorb`"conté tots els obstacles que s
acció $ObstaclesMolesten$(**ent** $c : Cami$)
var $o : Obstacle$; $f1, f2 : FST$ **fvar**
$f1 := ObrirFST(lectura,$ "`Planta`" $)$
$f2 := ObrirFST(escriptura,$ "`Destorb`" $)$
$LlegirObstacleFST(f1, o)$
mentre $\neg ObstSentinella(o)$ **fer**
si $Intercepta(o, c) \rightarrow EscriureObstacleFST(f2, o)$
$\square \ \neg Intercepta(o, c) \rightarrow \emptyset$
fsi
$LlegirObstacleFST(f1, o)$
fmentre
$TancarFST(f1)$
$TancarFST(f2)$
facció

{Prec: }
{Post: $Intercepta(o, c) \ \wedge \ o$ és interceptat per c}
funció $Intercepta$(**ent** $o : Obstacle$, **ent** $c : Cami$) **retorna booleà**
var $s : Segment$; $interc$: **booleà fvar**
$PrimerSegment(c, s)$;

```
  interc := fals
  mentre SegmentSentinella(s) ∧ ¬interc fer
    si ObstMolesta(o, s) → interc := cert
    □ ¬ObstMolesta(o, s) → SeguentSegment(c, s)
    fsi
  fmentre
  retorna interc
ffunció
```

El camí és una poligonal, és a dir, una seqüència de segments de línia recta que ve representada pel tipus *Cami*. Es demana que

1. Identifiqueu les mides del problema que influeixen sobre el cost en temps. En el cas d'haver cas millor i cas pitjor, identifiqueu aquest casos.

2. Determineu el cost asimptòtic en funció de la mides escollides per cadascun dels casos.

Nota: Totes les crides a accions i funcions no desenvolupades aquí tenen cost O(1).

2.2 Complexitat, exercici 3 de l'examen parcial del 3/03/2004

Donat el tipus

const Max : **enter** = 100000 **fconst**
tipus
T = **taula** [1..Max] **de enter**;
$taulaEnters$ = **tupla**
n : **enter**; {Nombre d'elements de la taula}
t : T
ftupla
ftipus

Considereu les següents funcions:

```
1  funció Quants1(ent ta, tb : taulaEnters)
2                    retorna enter
3    var
4      n, i : enter
5    fvar
7    si ta.n > 0 →
8        OrdenaSeleccio(tb)
9        n := 0; i := 1;
10       mentre (i ≤ ta.n) fer
11         si HiEsCercaBinaria(tb, ta.t[i]) →
12               n := n + 1
13         □ ¬HiEsCercaBinaria(tb, ta.t[i]) →
14         fsi
15         i := i + 1
16       fmentre
17   □ ta.n ≤ 0 →
18   fsi
19   retorna n
20 ffunció
```

```
1  funció Quants2(ent ta, tb : taulaEnters)
2                    retorna enter
3    var
4      n, i, j : enter; trobat : booleà
5    fvar
7    i := 1; n := 0
8    mentre i ≤ ta.n fer
9      trobat := fals; j := 1;
10     mentre (j ≤ tb.n ∧ ¬trobat) fer
11       si ta.t[i] ≠ tb.t[j] → j := j + 1
12       □ ta.t[i] = tb.t[j] → trobat := cert;
13                             n := n + 1
14       fsi
15     fmentre;
16     i := i + 1
17   fmentre
18   retorna n
19 ffunció
```

La funció $HiEsCercaBinaria$(**ent** t : $taulaEnters$, **ent** x : **enter**) realitza una cerca binària o dicotòmica dins de t per saber si x hi és a t. Retornarà cert en el cas de que hi sigui, fals en cas contrari. L'acció $OrdenaSeleccio$(**entsor** t : $taulaEnters$) fa l'ordenació de t pel mètode de selecció.

Es demana que :

a) Calculeu **raonadament** el cost assimptòtic dels algorismes en funció de la mida de les dades d'entrada n_a ($ta.n$) i n_b ($tb.n$). Si trobeu més d'un cas, suposeu el cas pitjor.

b) Analitzeu en quines condicions (sobre la mida de les dades d'entrada, i seguint sense tenir en compte les constants) és millor $Quants1$ que $Quants2$.

2.3 Complexitat04, exercici 3 de l'examen parcial del 2/10/2004

Suposem que definim el tipus següent:

```
1 tipus
2   taulaEnters = taula [1..100000] de enter
3 ftipus
```

Considereu el següent subprograma:

```
1  {Prec: 1 ≤ n ≤ 100000}
2  funció Nhiha(ent t : taulaEnters, ent n : enter) retorna booleà
4    var
5      i, j, sum : enter
6      trobat : booleà
7    fvar
9    i := 1;
10   trobat := fals
11   mentre i ≤ n ∧ ¬trobat fer
12     j := 1
13     sum := 0
14     mentre j ≤ i − 1 fer
15       sum := sum + t[j]
16       j := j + 1
17     fmentre
19     si
20       sum = t[i] → trobat := cert
21     □ sum ≠ t[i] → i := i + 1
22     fsi
23   fmentre
24   retorna trobat
25 ffunció
```

Es demana:

a) Calcular la complexitat asimptòtica en el cas pitjor d'aquest algorisme.

b)Modificar l'algorisme, sense variar l'especificació, a fi que la seva complexitat es redueixi, i calculeu-la.

2.4 ComplexitatIntersConjunts, exercici 3 de l'examen parcial del 1/04/2005

Ens proporcionen el tipus conjunt i el següent algorisme que l'utilitza:

```
1 tipus
2   conjunt = tupla
3               nElems : enter;   {Cardinalitat del conjunt}
4               elems : taulaEnters  {conté els elements del conjunt}
5             ftupla
6   taulaEnters = taula [1..100000] de enter
7 ftipus
```

```
1  {Pre: a = A i b = B i A no conté elements repetits i B tampoc}
2  funció nElemsInterseccioConjunts(ent a : conjunt, ent b : conjunt) retorna enter
3    var
4      i, j, n : enter
5      trobat : booleà
6    fvar
7    n := 0;
8    i := 1;
9    mentre i ≤ a.nElems fer
10     trobat := fals
11     j := 1
12     mentre j ≤ b.nElems ∧ ¬trobat fer
13       si a.elems[i] = b.elems[j] → trobat := cert; n := n + 1
14       □ a.elems[i] ≠ b.elems[j] → j := j + 1
15       fsi
16     fmentre
17     i := i + 1
18   fmentre
19   retorna n
20 ffunció
21   {Post: nElemsInterseccioConjunts(a, b) és el nombre d'elements d'A ∩ B}
```

Es demana:

a) Calculeu la complexitat asimptòtica en el cas pitjor d'aquest algorisme en funció del nombre d'elements d'A, n, i el nombre d'elements de B, m.

b) Sense canviar la postcondició i afegint alguna condició més a la precondició, és possible millorar la seva complexitat? En cas afirmatiu indiqueu la nova condició, en quin algorisme conegut es basaria la nova solució i també la nova complexitat resultant.

2.5 ComplexitatBombolla, exercici 3 de l'examen parcial del 2/10/2005

Ens proporcionen el tipus *tauEnters* i el següent algorisme que l'utilitza:

```
tipus
  tauEnters = tupla
                l : enter;   {La taula està ocupada entre les posicions 1 i l. (1 ≤ l ≤ 100000)}
                t : taulaEnters
              ftupla
  taulaEnters = taula [1..100000] de enter
ftipus
```

```
1  acció ordena(entsor te : tauEnters)
2    var
3      i, j : enter
4    fvar
5     {Pre: te = TE}
6    i := te.l;
7    mentre i > 1 fer
8      j := 2
9      mentre j ≤ i fer
10       si te.t[j − 1] > te.t[j] →
11                                    aux := te.t[j − 1]
12                                    te.t[j − 1] := te.t[j]
13                                    te.t[j] := aux
14       □ te.t[j − 1] ≤ te.t[j] →
15       fsi
16       j := j + 1
17     fmentre
18     i := i − 1
19   fmentre
20    {Post: te és una permutació dels elements de la taula t de TE, tal que es cumpleix te.t[i] ≤ t[j] per
      tot 1 ≤ i < j ≤ TE.l i te.l = TE.l}
21 facció
```

Identifiqueu la mida de les dades i calculeu la complexitat asimptòtica d'aquest algorisme. S'ha de presentar tot el desenvolupament que ens porta als resultats.

2.6 ComplexitatMultPolinomis, exercici 3 de l'examen parcial del 0/04/2006

Ens proporcionen la següent funció que utilitza el tipus *Polinomi*:

```
1  funció MultiplicaPolinomis(ent p, q : Polinomi) retorna Polinomi
2    var
3      m : Polinomi
4      gp, gq : enter
5      i, j : enter
6    fvar
7    m := CreaPolinomi()
8    i := 0
9    gp := Grau(p)
10   gq := Grau(q)
11   mentre i ≤ gp fer
12     j := 0
13     mentre j ≤ gq fer
14       AssignaCoef(m, i + j, Coef(p, i) ∗ Coef(q, i) + Coef(m, i + j))
15       j := j + 1
16     fmentre
17     i := i + 1
18   fmentre
19   retorna m
20 ffunció
```

{Prec: **cert**}
funció $CreaPolinomi()$ **retorna** $Polinomi$
{Post: $CreaPolinomi()$ és el polinomi $p(x) = 0$}

{Prec: $p(x) = a_0 + a_1x + \ldots + a_ix^i + \ldots + a_nx^n \ \wedge \ i \geq 0$}
acció $AssignaCoef$(**entsor** $p : Polinomi$, **ent** i : **enter**, **ent** c : **real**)
{Post: $p(x) = a_0 + a_1x + \ldots + cx^i + \ldots + a_nx^n$}

{Prec: **cert**}
funció $Grau$(**ent** $p : Polinomi$) **retorna enter**
{Post: $Grau(p)$ és el grau del polinomi, o sigui, l'índex del coeficient
no nul més gran de $p(x)$}

{Prec: $p(x) = a_0 + a_1x + \ldots + a_ix^i + \ldots + a_nx^n \ \wedge \ i \geq 0$}
funció $Coef$(**ent** $p : Polinomi$, **ent** i : **enter**) **retorna real**
{Post: $Coef(p, i) = a_i$}

Identifiqueu la mida de les dades i calculeu la complexitat asimptòtica d'aquest algorisme sabent que $CreaPolinomi$, $Coef$, $AssignaCoef$ tenen un cost en temps constant, i $Grau$ té un cost en temps lineal en funció del grau del polinomi.

S'ha de fer tot el desenvolupament que us porta als resultats.

2.7 AssignaXifra, exercici 3 de l'examen parcial del 3/10/2006

Donat

```
{Representació d'enter gran on només guardem xifres diferents de zero}
enterGran = tupla
              negatiu : booleà;
              nTermes : enter;
              t : tauTermes
            ftupla
tauTermes = taula [1..35] de terme;   {Taula ordenada de termes segons pes}
terme = tupla
          pes,
          xifra : enter
        ftupla
 1 acció AssigXifra(entsor e : enterGran, ent i : enter, ent x : enter)
 2   var pos : enter; trobat : booleà fvar
 3   CercaXifra(e, i, trobat, pos)
 4   si x = 0.0 →
 5                 si trobat → EsborraXifra(e, pos)
 6                 □ ¬trobat →
 7                 fsi
 8   □ x ≠ 0.0 →
 9                 si trobat → e.t[pos].xifra := x;
10                 □ ¬trobat → InsereixXifra(e, pos, i, x)
11                 fsi
12   fsi
13 facció
```

acció $CercaXifra$(**ent** $e : enterGran$, **ent** i : **enter**, **sor** $trobat$: **booleà**, **sor** pos : **enter**)
{**Pre:** $e = E$ i $ordenat(E)$ i $i = I$}
{**Post:** $trobat \rightarrow$ (E conté una xifra de pes i diferent de zero que està a la posició pos en la taula. $\neg trobat \rightarrow$ Si volem posar una xifra diferent de zero de pes i s'hauria de posar en pos, de forma que es mantinguin totes les xifres ordenades segons el pes de menor a major en la taula.}

acció $InsereixXifra$(**entsor** $e : enterGran$, **ent** pos : **enter**, i : **enter**, **ent** x : **real**)
{**Pre:** $e = E$ i $ordenat(E)$ i $i = I$ i $x = X$}
{**Post:** e conté a més de les xifres d'E, una altra xifra X de pes I, i totes les xifres estan ordenades de menor a major pes en una taula}

acció $EsborraXifra$(**entsor** $e : enterGran$, **ent** pos : **enter**)
{**Pre:** $e = E$ i $pos = POS$}
{**Post:** e té una xifra diferent de zero menys que E que està a la posició POS d'E}

On $ordenat(e) \equiv$ els termes d'$e.t$ estan ordenats de menor a major pes
Es vol analitzar la complexitat algorísmica de l'acció $AssigXifra$:

a) Identifiqueu la mida de les dades

b) Calculeu el cost asimptòtic en temps d'aquesta acció sabent que $CercaXifra$ és d'ordre logarítmic, $InsereixXifra$ és d'ordre lineal, i $EsborraXifra$ és d'ordre lineal, respecte la mida de les dades del problema.

S'ha de fer tot el desenvolupament que us porta als resultats.

3 Problemes d'estructures de dades

3.1 Expressio

Dissenyeu un algoritme tal que, donada una expressió, indiqui si és una expressió ben parentitzada. A l'expressió hi pot haver parèntesis, '(', ')', claus, '{', '}' i claudàtors '[', ']' i s'acaba en un punt, '.'. Es recomana usar el tipus *PilaCaracter* en el disseny.

Especificació tipus PilaCaracter

{Prec: **cert**}
acció $CrearPilaCaracter$(**sor** $p : PilaCaracter$)
{Post: $p = \emptyset$}
{Prec: $p = P$}
acció $EmpilarCaracter$(**entsor** $p : PilaCaracter$, **ent** e : **caràcter**)
{Post: $p = e.P$}
{Prec: $p = e.P$}
funció $CimPilaCaracter$(**ent** $p : PilaCaracter$) **retorna caràcter**
{Post: $e = CimPilaCaracter$}
{Prec: $p = e.P$}
acció $DesempilarCaracter$(**entsor** $p : PilaCaracter$)
{Post: $p = P$}
{Prec: **cert**}
funció $BuidaPilaCaracter$(**ent** $p : PilaCaracter$) **retorna booleà**
{Post: $BuidaPilaCaracter(p) \equiv p = \emptyset$}

3.2 Natalitat, exercici 4 de l'examen final del 2/06/2002

Un FST conté una seqüència d'enters acabats en -1 que corresponen al nombre de fills d'una mostra de dones de Catalunya.

Dissenyeu un algorisme que representi gràficament mitjançant un histograma, com el de la figura, les dades corresponents al nombre de catalanes que tenen 0, 1, ..., 9 fills. Podeu suposar que no hi ha cap dona amb més de 9 fills i que el grup del que hem pres la mostra és de 80 dones. Cal que useu una taula de freqüències en el disseny de l'algorisme. Calculeu també la mitjana del nombre de fills d'aquesta mostra i la variança.

```
0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1 XXXXXXXXXXXXXXX
2 XXXXXXXXXXXXXXXXXXXXX
3 XXXXXXX
4 XXX
5 X
6
7 XX
8
9 X
```

Podeu usar els subprogrames:
acció *EscriureCaracter*(**ent** c : **caràcter**)
acció *EscriuSaltLinia*()

NOTA: La variança de n nombres $x_1, x_2, \ldots, x_n$ és: $\sum_{i=1}^{n}(x_i - \overline{x})^2/(n-1)$ on $\overline{x}$ és la mitjana.

3.3 Cavalls, exercici 4 de l'examen final del 2/06/2003

Exercici 2: Curses de cavalls (3 punts)

Hem anat recomptant en una taula de freqüències el nombre de curses que ha guanyat cadascun dels cavalls d'un grup de 15. En una segona taula de freqüències, hem recomptat el nombre de curses perdudes (o sigui, les no guanyades) d'aquests cavalls. Els cavalls els identifiquem per un número entre 1 i 15.

Per poder valorar les apostes, volem calcular les probabilitats que té cadascun dels cavalls de guanyar l'altre (odds ratio): si el cavall A ha corregut T_A curses i d'aquestes n'ha guanyat G_A i el cavall B ha corregut T_B curses guanyant-ne G_B, la probabilitat que el cavall A guanyi al B la calcularem com $r_{AB} = (G_A/T_A)/(G_B/T_B)$. Si un cavall X no ha corregut cap cursa, per convenció es pren el valor $G_X/T_X = EPS$ a l'hora de fer aquest càlcul. EPS és una constant ja definida i $EPS > 0.0$

Dissenyeu un subprograma que a partir dels valors de les taules de freqüències, emmagatzemi en una matriu aquestes probabilitats. Cal que utilitzeu els tipus Matriu i TauFreq donats.

3.4 Imatges, exercici 4 de l'examen final del 2/06/2003

Disposem d'una seqüència d'imatges 3D que representen l'evolució d'una substància d'un determinat material que s'ha injectat en un fèmur d'un conill per experimentació. Cada imatge de la seqüència és una matriu 3D on cada element conté un valor real, que representa la propietat del material en estudi. Totes les matrius tenen les mateixes mides en les tres dimensions

Aquesta seqüència d'imatges 3D està en un fitxer de nom "Imatges.txt", on primer hi ha les tres mides comuns a totes les matrius, m, n i p, i després hi ha totes les matrius 3D de la seqüència, acabades amb una matriu sentinella de les mateixes dimensions: I_1, I_2, ..., I_n,I_S.

Com que les matrius d'aquesta seqüència són molt semblants entre si, es vol aplicar un mètode de compactació que consisteix en calcular i guardar les diferències entre matrius consecutives.

Es defineix $dif_{t,t+1}$ com la diferència entre les imatges I_t i I_{t+1} i $dif_{t,t+1}$ conté el nombre d'elements diferents entre les imatges I_t i I_{t+1} i, per cada element, conté la seva posició (i, j, k) i el valor a la imatge I_{t+1}.

Aquest mètode de compactació s'usarà per obtenir un fitxer comprimit de nom "ImatgesComprimides.txt" on hi hagi les tres mides m, n i p, la matriu corresponent a la primera imatge completa, I_1, i les successives diferències $dif_{1,2}$, $dif_{2,3}$, ..., $dif_{n-1,n}$

Es demana:

1. Especifiqueu i dissenyeu un subprograma que calculi la diferència (tal com s'ha definit en aquest problema) entre dues matrius.

2. Dissenyeu un algorisme que obtingui el fitxer "ImatgesComprimides.txt" a partir del fitxer "Imatges.txt", usant el subprograma de l'anterior apartat

Es disposa de:

- Un tipus $Matriu3D$ anàleg al tipus $Matriu$ conegut.

- La següent funció, que ens diu si una matriu és una matriu Sentinella:
  {Prec: }
  funció $MatriuSentinella$(**ent** $a : Matriu3D$) **retorna booleà**
  {Post: retorna cert si a és la matriu sentinella}

- Un tipus $Diferencia$ amb les següents operacions:
  {Prec: }
  funció $CrearDiferencia()$ **retorna** $Diferencia$
  {Post: $CrearDiferencia()$ és una variable de tipus Diferencia buida}
  {Prec: $d = D = d_1, d_2, ..., d_n \ \wedge \ n = N \ \wedge \ N \geq 0$}
  acció $AfegeixDif$(**entsor** $d : Diferencia$, **ent** i, j, k : **enter**, **ent** $valor$: **real**)
  {Post: $d = D.d_{N+1} \ \wedge \ d_{N+1} = (i, j, k, valor)$}
  {Prec: $mode(f) = escriptura \ \wedge \ pe(f) = \alpha$}
  acció $EscriureDifFST$(**entsor** $f : FST$, **ent** $d : Diferencia$)
  {Post: $mode(f) = escriptura \ \wedge \ pe(f) = \alpha.d$}

3.5 Votacions, exercici 4 de l'examen final del 2/01/2004

Dissenyeu un algorisme per facilitar l'escrutini dels vots d'una mesa electoral a les Eleccions per al Parlament de Catalunya. Utilitzeu una taula de freqüències. Suposareu que només hi participen cinc partits: Convergència i Unió s'anomena 'C', ERC 'E', Iniciativa-Verds 'I', el PP 'P' i el PSC 'S'. Un vot en blanc és 'B' i un de nul 'N'. Cada vot es codifica mitjançant una d'aquestes lletres. El final de la seqüència de vots s'indica amb la lletra 'F'. Si apareix un caràcter que no és cap dels mencionats fins aqui, es compta com a vot nul. La seqüència de caràcters es llegeix des d'un FST que es diu "vots.txt". L'algorisme que dissenyareu ha de:

- Calcular els percentatges de vots vàlids de cada partit, és a dir, sobre el total de la suma de vots dels cinc partits (o sigui, exclosos els blancs i els nuls; i la F tampoc no compta dins el total de vots).
- Escriure total de vots i percentatge de cadascun dels cinc partits; i, a continuació, total de blancs i total de nuls.

Podeu utilitzar l'especificació adjunta.

3.6 EnquestesTrens, exercici 4 de l'examen final del 2/06/2004

La RENFE està fent unes enquestes per conèixer informació sobre els tipus de desplaçaments que fan els usuaris a la línia de la costa. Les persones enquestades han d'indicar l'estació d'orígen i la de destinació. Les estacions estan agrupades per zones i RENFE vol fer una classificació de l'ús del seu servei basat en el nombre de zones travessades (NZA). A la regió on es fan aquestes enquestes hi ha 10 zones.

Es disposa d'un fitxer de nom "enquestes.txt" amb una seqüència d'enquestes. Cada enquesta consta de dues dades corresponents a l'estació origen i la destinació (de tipus *tEstació*). La seqüència acaba amb dues estacions sentinelles.

La RENFE vol disposar, en un fitxer de nom "resultats.txt", de la següent informació corresponent a la línia enquestada: una llista de parelles (NZA, nombre de persones), per cadascún dels possibles valors de NZA (entre 1 i 10, ordenats creixentment, i apareixeran sempre encara que el nombre de persones sigui zero), el NZA promig i el NZA amb el nombre màxim de persones (moda).

Es disposa de les funcions que treballen sobre el tipus $tEstacio$:

funció $NZAentreEst$(**ent** $eo, ed : tEstacio$) **retorna enter**
{Pre: $eo = EO$ i $ed = ED$}
{Post: $NZAentreEst(ep, ed)$ és el nombre de zones que hi ha entre les estacions EO i ED}
acció $LlegirEstacioFST$(**entsor** $f : FST$, **sor** $e : tEstacio$)
{Pre: $mode(f) = lectura$ i $\neg fdf(f)$ i $pd(f) = \epsilon \cdot \gamma$ i $pe(f) = PE$}
{Post: e representa ϵ. $pd(f) = \gamma$, $pe(f) = PE \cdot \epsilon$}
funció $EstacioSent$(**ent** $e : tEstacio$) **retorna booleà**
{Pre: }
{Post: $EstacioSent(e)$ és cert si i sols si e és l'estació sentinella}

Dissenyeu un algorisme que obtingui l'esmentada informació.

Cal que useu el tipus Taula de Freqüències del qual s'adjunta l'especificació.

3.7 Verbs, exercici 4 de l'examen final del 1/01/2005

El fitxer de text `"verbs.txt"` conté una seqüència de paraules, on cada paraula correspon a l'arrel d'un verb català. Altrament dit, conté les arrels de tots els verbs catalans. El fitxer de text `"tirant.txt"` conté una seqüència de paraules corresponent a la noveł.la *Tirant lo Blanc*. Donats els dos fitxers anteriors, dissenyeu un algorisme que escrigui en el fitxer de text `"freq.txt"` el nombre de vegades que apareix cada verb català en el *Tirant*. El fitxer `"freq.txt"` ha de contenir parelles (verb, freqüència) on el verb també està en `"verbs.txt"` i ha aparegut com a mínim un cop en el *Tirant*. L'escriptura en el fitxer `"freq.txt"` s'ha de fer amb l'ordre de verbs que hi ha al fitxer `"verbs.txt"`.

Disposeu dels tipus i subprogrames següents:

tipus *Paraula* **ftipus**

acció $LlegirParaulaFST$(**entsor** $f : FST$, **sor** $p : Paraula$)
{**Pre:** f obert en mode lectura $\wedge$ queda alguna paraula per llegir}
{**Post:** p és la següent paraula en f}
funció $ParaulaSentinella$(**ent** $p : Paraula$) **retorna booleà**
{**Pre: cert**}
{**Post:** $ParaulaSentinella(p)$ = **cert** si i només si p és la paraula sentinella}
funció $EsVerbParaula$(**ent** $p : Paraula$) **retorna booleà**
{**Pre: cert**}
{**Post:** $EsVerbParaula(p)$ = **cert** si i només si p és un verb}
funció $ArrelParaula$(**ent** $p : Paraula$) **retorna** $Paraula$
{**Pre: cert**}
{**Post:** $ArrelParaula(p)$ és l'arrel de p}
acció $EscriureParaulaFST$(**entsor** $f : FST$, **ent** $p : Paraula$)
{**Pre:** f obert per escriptura}
{**Post:** Escriu al fitxer f la paraula p}

tipus $TauFreq$ **ftipus**
acció $Inicialitzar$(**sor** $t : TauFreq$)
{**Pre: cert**}
{**Post:** t és una taula de freqüències buida}
acció $Afegir$(**entsor** $t : TauFreq$, **ent** $x : Paraula$)
{**Pre:** $Freq(t, x) = F$}
{**Post:** $Freq(t, x) = F + 1$ $\wedge$ la freqüència en t de la resta d'elements no ha canviat}
funció $Freq$(**ent** $t : TauFreq$, **ent** $x : Paraula$) **retorna enter**
{**Pre: cert**}
{**Post:** $Freq(t, x)$ és la freqüència d'x en t}

El tipus *TauFreq* **NO** té cap més subprograma que els anteriors (llevat dels que especifiqueu i dissenyeu vosaltres).

3.8 Bicicletes, exercici 4 de l'examen final del 1/06/2005

L'ajuntament està fent un estudi sobre el tràfic existent en els carrils-bici de la ciutat. Per fer-ho, s'han instal·lat sensors en diversos punts geogràfics de la ciutat. Quan un sensor detecta el pas d'una bicicleta, ho notifica a un ordinador central. Al final del dia es té un fitxer text "senyals.txt" consistent en una seqüència de paraules que identifiquen indrets geogràfics i que es repeteixen en la seqüència cada cop que ha passat una bicicleta en el lloc geogràfic en qüestió.

Ens demanen fer un programa que donat el fitxer text "senyals.txt" i un fitxer text anomenat "carrilDiagonal.txt" que conté només els identificadors d'indrets geogràfics que pertanyen al carril bici de la Diagonal, tregui pel canal standard de sortida la mitjana de tràfic del carril de la Diagonal, i quin indret d'aquest carril hi ha més tràfic de bicicletes.

Disposeu dels tipus i subprogrames següents:

tipus $Paraula$ **ftipus**
acció $LlegirParaulaFST$(**entsor** $f : FST$, **sor** $p : Paraula$)
{**Pre:** f obert en mode lectura $\wedge$ queda alguna paraula per llegir}
{**Post:** p és la següent paraula en f}
funció $ParaulaSentinella$(**ent** $p : Paraula$) **retorna booleà**
{**Pre: cert**}
{**Post:** $ParaulaSentinella(p)$ = **cert** si i només si p és la paraula sentinella}

3.9 CentreCalcul, exercici 4 de l'examen final del 1/01/2006

Un centre de càlcul disposa de dos servidors als quals es connecten els seus usuaris. Cada vegada que un usuari es connecta al servidor 1, s'enregistra la data, l'hora i l'identificador de l'usuari al fitxer "login-1.log". El fitxer "login-2.log" conté la mateixa informació, però per al servidor 2. Els identificadors d'usuari són enters entre 1000 i 10000. Els fitxers "login-1.log" i "login-2.log" acaben amb un element que conté una data i hora arbitràries i un identificador d'usuari igual a -1. Per tal de millorar el servei, els responsables del centre de càlcul volen saber el nombre de vegades que s'ha connectat cada usuari en un interval de dies.

Es demana que dissenyeu un algorisme que donat un interval de dies pel Canal Estàndard d'Entrada (CSE) i donats els fitxers "login-1.log" i "login-2.log", escrigui pel Canal Estàndard de Sortida (CSS) una seqüència de parelles formada per un identificador d'usuari i un enter que indica el nombre de vegades que ha entrat al sistema aquest usuari, és a dir, el nombre de vegades que ha entrat al servidor 1 més el nombre de vegades que ha entrat al servidor 2. Només s'han d'escriure els usuaris que han entrat un cop com a mínim. Per exemple, per l'interval de temps del 2/1/2006 al 9/1/2006, i la resta de dades següents:

CSE	fitxer "login-1.log"	fitxer "login-2.log" és
2/1/2006	2/1/2006 10:30 1024	4/1/2006 21:33 2048
9/1/2006	2/1/2006 12:43 5734	6/1/2006 10:21 1024
	3/1/2006 19:28 1024	12/1/2006 8:43 2048
	5/1/2006 14:01 1024	13/1/2006 9:12 -1
	7/1/2006 17:25 2048	
	10/1/2006 8:20 1024	
	11/1/2006 2:12 -1	

el resultat que ha d'escriure l'algorisme al CSS és 1024 4 2048 2 5734 1

Es disposa dels tipus $Data$, $Hora$ i $TauFreq$, i dels subprogrames següents:

acció $LlegirData$(**sor** $d : Data$)[1]

acció $LlegirDataFST$(**entsor** $f : FST$, **sor** $d : Data$)[2]

acció $LlegirHoraFST$(**entsor** $f : FST$, **sor** $h : Hora$)[2]

funció $DinsInterval$(**ent** $ii : Data$, **ent** $if : Data$, **ent** $d : Data$) **retorna booleà**

{**Pre: cert**}

{**Post:** $DinsInterval(ii, if, d)$ = **cert** si la data d està entre ii i if inclosos.}

acció $Inicialitzar$(**sor** $t : TauFreq$)

{**Pre: cert**}

{**Post:** t és una taula de freqüències buida}

acció $Afegir$(**entsor** $t : TauFreq$, **ent** uid : **enter**)

{**Pre:** $1000 \leq uid \leq 10000$ i $Freq(t, uid) = F$}

{**Post:** $Freq(t, uid) = F + 1 \ \wedge$ la freqüència en t de la resta d'uids no ha canviat}

funció $Freq$(**ent** $t : TauFreq$, **ent** uid : **enter**) **retorna enter**

{**Pre:** $1000 \leq uid \leq 10000$}

{**Post:** $Freq(t, x)$ és la freqüència d'x en t}

[1] Precondicions i Postcondicions similars a LlegirEnter, canviant **enter** pel tipus Data o Hora.

[2] Precondicions i Postcondicions similars a LlegirEnterFST, canviant **enter** pel tipus Data o Hora.

3.10 Magatzem, exercici 4 de l'examen final del 1/06/2006

Un gran magatzem controla l'entrada i sortida dels productes que guarda mitjançant l'enregistrament del seu codi de barres. Per actualitzar el seu inventari disposa dels següents FSTs: El FST "entrada" conté una seqüència de codis de barres, un codi per cada unitat de producte que ha entrat al magatzem des del darrer inventari efectuat. El FST "sortida" conté una seqüència de codis de barres, un codi per cada unitat de producte que ha sortit

del magatzem des del darrer inventari efectuat. Tant el FST "entrada" com el de "sortida" acaben amb un codi de barres sentinella. El FST "inventari" conté una seqüència de parelles (c, n) on c és el codi de barres d'un producte i n el nombre de productes del codi de barres associat que hi havia al magatzem abans d'efectuar-se les entrades i sortides. Aquest darrer FST conté els codis de barres de tots els possibles productes que poden entrar o sortir del magatzem i acaba amb una parella en que el seu codi de barres és el mateix sentinella que pel cas del FST "entrada".

Es demana dissenyar un algorisme que a partir dels FST "entrada", "sortida", i "inventari" descrits abans, construeixi un altre FST "NouInventari" que contingui els codis de barres amb el mateix ordre que el FST "inventari" juntament amb el nous nombres de productes resultants de les entrades i sortides efectuades. El nombre de parelles del FST "NouInventari" serà el mateix que el del FST "inventari".

acció $LlegirCodiBarresFST$(**entsor** $f : FST$, **sor** $cb : codiBarres$)[1]
funció $CodiBarresSentinella$(**ent** $c : codiBarres$) **retorna booleà**
{**Pre: cert**}
{**Post:** $CodiBarresSentinella(c)$ = **cert** si i només si c és el codi de barres sentinella}
acció $EscriureCodiBarresFST$(**entsor** $f : FST$, **ent** $cb : codiBarres$)
{**Pre:** f obert per escriptura}
{**Post:** Escriu al fitxer f ek cidu de barres cb}
acció $Inicialitzar$(**sor** $t : TauFreq$)
{**Pre: cert**}
{**Post:** t és una taula de freqüències buida}
acció $Afegir$(**entsor** $t : TauFreq$, **ent** $cb : codiBarres$)
{**Pre:** $Freq(t, cb) = F$}
{**Post:** $Freq(t, cb) = F + 1 \ \wedge$ la freqüència en t de la resta de codis de barres no ha canviat}
funció $Freq$(**ent** $t : TauFreq$, **ent** $cb : codiBarres$) **retorna enter**
{**Pre: cert**}
{**Post:** $Freq(t, cb)$ és la freqüència d'cb en t}

[1] Precondicions i Postcondicions similars a LlegirEnterFST, canviant **enter** pel tipus codiBarres.

3.11 ControlVersions, exercici 4 de l'examen final del 2/01/2007

Un sistema de control de versions permet conservar la història de les modificacions que un conjunt d'usuaris fa sobre el conjunt de fitxers que constitueixen un projecte. Amb l'objectiu de facilitar càlculs estadístics, el sistema de control de versions enregistra en un fitxer per cada projecte, la data, l'usuari i el nom del fitxer cada vegada que un usuari modifica un dels fitxers que constitueixen el projecte. En el nostre cas, el sistema de control de versions gestiona dos projectes i enregistra les modificacions als fitxers `"hibrids.log"` i `"justificacio.log"`. Ambdós fitxers estan ordenats creixentment per la data de modificació. Cada element d'ambdós fitxers està format per una data (tres enters corresponents a l'any, mes i dia), un nom d'usuari i un nom de fitxer. El sentinella dels dos fitxers és un element amb una data sentinella amb any, mes i dia iguals a zero.

A partir de les dades contingudes als fitxers `"hibrids.log"` i `"justificacio.log"`, ens demanen que el programa escrigui al canal estàndard de sortida, primer el contingut dels dos fitxers ordenat per data, i a continuació quantes modificacions de fitxers s'han produït en cada data. Només s'han d'escriure les dates en les quals s'ha modificat algun fitxer.

Per exemple,

si el fitxer `"hibrids.log"` conté les dades següents:

```
2006 10 3 jaume paraula.h
2006 10 3 pere llistanoms.c
2006 10 12 maria llistanoms.h
2006 10 15 pau entsort.c
2006 10 15 ernest paraula.h
2006 10 23 joana llistanoms.c
2006 11 7 jaume llistanoms.h
2006 11 7 jaume entsort.c
0 0 0 xxx zzz
```

I el fitxer `"justificacio.log"` conté les dades següents:

```
2006 10 5 jaume paraula.h
2006 10 12 maria linia.h
2006 10 14 pau entsort.c
2006 10 15 ernest paraula.h
0 0 0 xxx zzz
```

Cal que el programa escrigui:

```
2006 10 3 jaume paraula.h
2006 10 3 pere llistanoms.c
2006 10 5 jaume paraula.h
2006 10 12 maria llistanoms.h
2006 10 12 maria linia.h
2006 10 14 pau entsort.c
2006 10 15 pau entsort.c
2006 10 15 ernest paraula.h
2006 10 15 ernest paraula.h
2006 10 23 joana llistanoms.c
2006 11 7 jaume llistanoms.h
2006 11 7 jaume entsort.c
2006 10 3 2
2006 10 5 1
2006 10 12 2
2006 10 14 1
2006 10 15 3
2006 10 23 1
2006 11 7 2
```

Tingueu en compte que:

- Els fitxers `"hibrids.log"` i `"justificacio.log"` contenen un element com a mínim diferent del sentinella.
- Les parelles formades per una data i el nombre de modificacions de fitxers corresponents a aquesta data s'han d'escriure al canal estàndard de sortida ordenades temporalment.
- Disposeu del tipus *Data* i dels subprogrames següents:

  acció *LlegirDataFST*(**entsor** f : *FST*, **sor** *data* : *Data*)
  acció *EscriureData*(**ent** d : *Data*)
  funció *DataSentinella*(*data* : *Data*) **retorna booleà**
  funció *CompararData*(**ent** $d1, d2$: *Data*) **retorna enter**
  {Prec: **cert**}
  {Post: retorna x on $x = 0$ si $d1 = d2$, $x < 0$ si $d1 < d2$, $x > 0$ si $d1 > d2$}

  funció *SegüentData*(**ent** d : *Data*) **retorna** *Data*
  {**Pre: cert**}
  {**Post:** retorna el dia següent de d. Per exemple, si d representa el 31 de Gener del 2007, *SegüentData* retorna el que representa l'1 de Febrer del 2007}

- Disposeu del tipus *Nom* i dels subprogrames següents:

  acció *LlegirNomFST*(**entsor** f : *FST*, **sor** n : *Nom*)
  acció *EscriureNom*(**ent** n : *Nom*)

- Disposeu del tipus *TauFreq* i dels subprogrames següents:

  acció *Inicialitzar*(**sor** t : *TauFreq*)
  {**Pre: cert**}
  {**Post:** t és una taula de freqüències buida}
  acció *Afegir*(**entsor** t : *TauFreq*, **ent** d : *Data*)
  {**Pre:** $Freq(t, d) = F$}
  {**Post:** $Freq(t, d) = F + 1 \ \wedge$ la freqüència en t de la resta de dates no ha canviat}
  funció *Freq*(**ent** t : *TauFreq*, **ent** d : *Data*) **retorna enter**
  {**Pre: cert**}
  {**Post:** $Freq(t, d)$ és la freqüència de la data d en t}

4 Problemes de càlcul numèric

4.1 MaxValorPropi

Construïu una aplicació per obtenir el valor propi més gran d'una certa matriu. La matriu $A(n \times n)$ s'obtindrà pel CEE considerant el format següent:

$$< n >< fila_1 > \cdots < fila_n >$$

Per calcular el valor propi major farem servir la seqüència següent de X_k i Z_k:

$$Z_k = AX_k$$

$$X_{k+1} = Z_k / \mid Z_k \mid$$

tal que si prenem X_0 com qualsevol vector, llavors el valor $\mid Z_k \mid / \mid X_k \mid$ tendeix al valor propi més gran i X_k al vector propi corresponent. Considereu que el mètode ha convergit quan dues aproximacions succesives del valor propi siguin suficientment properes.

4.2 ArrelPolinomi, exercici 5 de l'examen final del 1/01/2001

El mètode de Newton per trobar arrels d'una funció $f(x)$ és un mètode iteratiu que parteix d'una aproximació de la solució inicial x_0 i va fent iteracions obtenint noves aproximacions $x_{k+1} = x_k + f(x_k)/f'(x_k)$ fins que convergeix. Es considera que el mètode convergeix quan el valor de la funció en una aproximació és inferior a un valor *epsilon* donat. El que es demana és especificar i dissenyar un subprograma que obtingui una arrel d'una funció polinòmica aplicant aquest mètode. Se suposarà conegut el tipus *Polinomi* i s'haurà de fer ús de les seves operacions, l'especificació de les quals s'inclou al final de l'examen.

Cal que seguiu els següents passos:

1. Especifiqueu un subprograma de nom *Deriva* tal que donat un polinomi n'obtingui un altre que sigui la seva derivada.

2. Implementeu el subprograma *Deriva* fent ús de les operacions sobre el tipus *Polinomi*. Indiqueu quina és la seqüència a tractar i caracteritzeu-la. Indiqueu quin esquema feu servir.

3. Especifiqueu un subprograma de nom *Avalua* tal que donat un polinomi, P, i un valor real, x, calculi el valor $P(x)$.

4. Implementeu el subprograma *Avalua* aplicant el mètode de Horner: $P(x) = a_0 + (a_1 + (a_2 + (\ldots(a_n x)\ldots))x)x$ i fent ús de les operacions sobre el tipus Polinomi donades. Indiqueu quina és la seqüència a tractar i caracteritzeu-la. Indiqueu quin esquema feu servir.

5. Especifiqueu un subprograma *Newton* tal que donat un polinomi, P, una aproximació inicial d'una seva arrel, x_0, i un valor *epsilon*, calculi una arrel del polinomi aplicant el mètode de Newton explicat a l'inici de l'enunciat. Es pot suposar que les condicions inicials donades, x_0 i *epsilon* permeten que el mètode de Newton convergeixi pel polinomi donat P.

6. Implementeu el subprograma *Newton* usant els subprogrames *Deriva* i *Avalua*. Indiqueu quina és la seqüència a tractar i caracteritzeu-la. Indiqueu quin esquema feu servir.

7. Analitzeu la complexitat del subprograma *Deriva* en funció del grau del polinomi.

Especificació del tipus Polinomi

{Prec: }
funció *CrearPolinomi*(**ent** n : **enter**) **retorna** *Polinomi*
{Post: *CrearPolinomi*$(n) = p \ \wedge \ p$ és un polinomi de grau
n amb tots els coeficients nuls}

{Prec: $e \geq 0 \ \wedge \ e \leq n \ \wedge \ p = a_0 + a_1 x + \ldots + a_n x^n$}
acció *AfegeixTerme*(**entsor** p : *Polinomi*, **ent** e : **enter**, **ent** a : **real**)
{Post: $p = a_0 + a_1 x + \ldots + a x^e + \ldots + a_n x^n$}

{Prec: $p = a_0 + a_1 x + \ldots + a_n x^n \ \wedge$ }
funció *GrauPolinomi*(**ent** p : *Polinomi*) **retorna enter**
{Post: *GrauPolinomi*$(p) = n$}

{Prec: $e \geq 0 \ \wedge \ e \leq n \ \wedge \ p = a_0 + a_1 x + \ldots + a_e x^e + \ldots + a_n x^n$}
funció *CoeficientTerme*(**ent** p : *Polinomi*, **ent** e : **enter**) **retorna real**
{Post: *CoeficientTerme*$(p, e) = a_e$}

{Prec: }
funció *DestruirPolinomi*(**ent** p : *Polinomi*)
{Post: p destruït }

4.3 Quefa, exercici 5 de l'examen final del 2/06/2002

Un company ens passa el següent codi en C:

```
#define N 10

typedef int T[N];

int main(void)
{
  T t;
  fes_quelcom(t);
}

void fes_quelcom(T t)
{
  int *p, *d, a;

  p = &t[0];
```

```
  d = &t[N-1];
  while (p < d) {
    a = *p;
    *p = *d;
    *d = a;
    p = p + 1;
    d = d - 1;
  }
}
```

I ens demana que li diguem:

1. Quin serà el valor de `t` després de la crida `fes_quelcom(t)` si el valor inicial era `t= 0, 1, 2, 3, 4, 5, 6, 7, 8, 9` ?
2. Que fa l'acció `fes_quelcom(t)` ?

4.4 VectorPropi, exercici 5 de l'examen final del 2/06/2002

Especifiqueu i dissenyeu un subprograma que calculi el valor propi més gran d'una matriu quadrada A donada. Un mètode per calcular el valor propi més gran genera una seqüència de vectors X_k, Z_k de la forma:

$$Z_k = AX_k$$

$$X_{k+1} = Z_k/\|Z_k\|$$

i començant per un X_0 que pot ser qualsevol vector normalitzat. Aquest mètode ens garanteix que el valor $\|Z_k\|$ tendeix al valor propi més gran i X_k al vector propi corresponent. Considereu que el mètode ha convergit quan dues aproximacions successives del valor propi siguin suficientment properes.

Heu d'usar els tipus $Vector$ i $Matriu$. Podeu usar els següents subprogrames:
{Prec: $ColsMatriu(a) = DimVector(b)$}
acció $ProdMatVect$(**ent** $a : Matriu$, **ent** $b : Vector$, **sor** $c : Vector$)
{Post: $c = a * b$}
{Prec: $cert$}
funció $ModulVector$(**ent** $b : Vector$) **retorna real**
{Post: $ModulVector(b) = b$}

Es valorarà especialment la caracterització de la seqüència i l'aplicació de l'anàlisi descendent en el disseny del subprograma que es demana.

4.5 IntegralPolinomis, exercici 5 de l'examen final del 2/01/2004

Disposem de la següent definició del tipus Polinomi:

{Prec: **cert**}
funció $CreaPolinomi()$ **retorna** $Polinomi$
{Post: Retorna el polinomi $p(x) = 0$}

{Prec: $p(x) = a_0 + a_1x + \ldots + a_ix^i + \ldots + a_nx^n$}
acció $AssignaCoef($**entsor** $p : Polinomi,$ **ent** $i :$ **enter**, **ent** $c :$ **real**$)$
{Post: $p(x) = a_0 + a_1x + \ldots + cx^i + \ldots + a_nx^n$}

{Prec: **cert**}
funció $Grau($**ent** $p : Polinomi)$ **retorna enter**
{Post: Retorna el grau del polinomi}

{Prec: $p(x) = a_0 + a_1x + \ldots + a_ix^i + \ldots + a_nx^n$}
funció $Coef($**ent** $p : Polinomi,$ **ent** $i :$ **enter**$)$ **retorna real**
{Post: Retorna a_i}

Especifiqueu i implementeu un subprograma que a partir d'un polinomi $Q(x)$ calculi el valor de la seva integral en l'intèrval $[a, b]$, és a dir,

$$\int_a^b Q(x)\, dx$$

4.6 Regressio, exercici 5 de l'examen final del 2/06/2004

Volem ajustar una recta de regressió $y = ax + b$ a un conjunt de mostres $\{x_i, y_i\}, i = 1 \ldots n$. Les dades d'entrada provenen d'un fitxer de text anomenat `''punts.txt''`, el qual conté una seqüència de parelles de coordenades reals acabada amb el sentinella $-1, -1$.

Recordem que el mètode de la regressió lineal simple per mínims quadrats determina que

$$\begin{aligned} a &= \frac{\bar{x}\,\bar{y} - \frac{1}{n}\sum_{i=1}^{n} x_i y_i}{(\bar{x})^2 - \frac{1}{n}\sum_{i=1}^{n} x_i^2} \\ b &= \bar{y} - a\bar{x} \end{aligned}$$

essent $\bar{x}, \bar{y}$ les mitjanes $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Dissenyeu un algorisme que a partir de les dades del fitxer trobi els paràmetres que determinen la recta, i els escrigui al canal stàndard de sortida.

4.7 NucliMatriusv1, exercici 5 de l'examen final del 1/01/2005

Dissenyar un algorisme que donada una matriu quadrada A, $n \times n$, pel CEE i un FST, "vectors.dat" que conté vectors de dimensió n, construeixi un nou FST format pels vectors de "vectors.dat" que pertanyin al Nucli de la matriu A.

acció $LlegirVectorFST($**entsor** $f : FST,$ **sor** $v : Vector$**)**
{**Pre:** f obert en mode lectura $\wedge$ queda algun vector per llegir}
{**Post:** p és el següent vector en f}
funció $VectorSentinella($**ent** $v : Vector$**)** **retorna booleà**
{**Pre: cert**}
{**Post:** $VectorSentinella(v)$ és **cert** si i només si v és el vector sentinella}

acció $EscriureVectorFST$(**entsor** $f : FST$, **ent** $v : Vector$)
{**Pre:** f obert en mode escriptura}
{**Post:** v s'ha escrit al final d'f}
funció $ModulVector$(**ent** $v : Vector$) **retorna real**
{**Pre: cert**}
{**Post:** $ModulVector(v)$ és $\|v\|$}
acció $LlegirMatriu$(**sor** $M : Matriu$)
{**Pre: cert**}
{**Post:** M és una matriu $n \times n$ amb valors inicialitzats}

4.8 Tridiagonal, exercici 5 de l'examen final del 1/06/2005

Donada una matriu tridiagonal A, dissenyeu un subprograma que en faci l'eliminació gaussiana el més eficient possible, és a dir, fent el nombre mínim de sumes, multiplicacions i divisions.

Una matriu tridiagonal és una matriu en la qual els únics elements diferents de zero en cada fila (i) són l'element de la diagonal (a_{ii}) i els de les columnes anterior ($a_{i,i-1}$) i posterior ($a_{i,i+1}$):

$$\begin{array}{llllll} a_{00}x_0+ & a_{01}x_1 & & & & = b_0 \\ a_{10}x_0+ & a_{11}x_1+ & a_{12}x_2 & & & = b_1 \\ & a_{21}x_1+ & a_{22}x_2+ & a_{23}x_3 & & = b_2 \\ \multicolumn{6}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\ & & & & a_{n-1,n-2}x_{n-2} \quad +a_{n-1,n-1}x_{n-1} & = b_n \end{array}$$

Recordeu que l'eliminació gaussiana es calcula per $k = 0, 1, \ldots, n-2$, aplicant les fórmules següents:

$$\left.\begin{array}{l} a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} \cdot a_{kj}^{(k)} \\ m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \end{array}\right\} \begin{array}{l} \text{per } i = k+1, \ldots, n-1 \\ \text{i per } j = k+1, \ldots, n-1 \end{array}$$

Podeu suposar que $a_{kk}^{(k)} \neq 0$ sempre i que no cal aplicar pivotatge.

Noteu que aquestes fórmules són generals i es poden simplificar en el cas de les matrius tridiagonals.

4.9 NucliMatriusv3, exercici 5 de l'examen final del 1/01/2006

Dissenyeu un algorisme que donada una matriu quadrada M, $n \times n$, pel CEE i un FST, "vectors.dat" que conté vectors de dimensió n, indiqui si hi ha algun vector v_i, de "vectors.dat", tal que satisfà que l'expressió $v_i + M \cdot v_i$ pertanyi al Núcli de la matriu M. L'algorisme escriurà 'S' en el cas de que existeix l'esmentat vector v_i i 'N' en cas contrari. Es valorarà el disseny descendent plantejat.

A més de les accions i/o funcions que teniu en el formulari, també podeu disposar de:
acció $LlegirVectorFST$(**entsor** $f : FST$, **sor** $v : Vector$)
{**Pre:** f obert en mode lectura $\wedge$ queda algun vector per llegir}
{**Post:** p és el següent vector en f}
funció $VectorSentinella$(**ent** $v : Vector$) **retorna booleà**
{**Pre: cert**}
{**Post:** $VectorSentinella(v)$ és **cert** si i només si v és el vector sentinella}
acció $LlegirMatriu$(**sor** $M : Matriu$)
{**Pre: cert**}
{**Post:** M és una matriu $n \times n$ amb valors inicialitzats}

4.10 Cramer, exercici 5 de l'examen final del 1/06/2006

Es vol dissenyar una acció que resolgui un sistema de n equacions amb n incògnites mitjançant el mètode de Cramer amb la següent especificació:

acció $resolSistemaCramer$(**ent** $a : Matriu$, **ent** $b : Vector$, **sor** $x : Vector$)

{Prec: $FilesMatriu(a) = ColsMatriu(a) = DimVector(b) = DimVector(x)$ i $\mid a \mid \neq 0$}

{Post: x és tal que $a \cdot x = b$}

El mètode de Cramer, a partir d'un sistema d'equacions com

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{\mathbf{b}}$$

indica que cal fer el següent: Cada component x_i del vector $\mathbf{x}$ es calcula fent $x_i = \frac{|B_i|}{|A|}$ i B_i és una matriu quadrada $n \times n$ tal que tots els seus components excepte els de la columna i coincideixen amb els components de la matriu A, i els components que hi han a la columna i de B_i coincideixen amb els del vector de termes independents $\mathbf{b}$. És a dir,

$$B_i = \begin{pmatrix} a_{11} & \cdots & a_{1,i-1} & b_1 & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2,i-1} & b_2 & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & & \\ a_{n1} & \cdots & a_{n,i-1} & b_n & \cdots & a_{nn} \end{pmatrix}$$

Dissenyeu l'acció $resolSistemaCramer$ que porti a terme el mètode exposat tenint en compte que disposeu de la següent funció:

funció $Determinant$(**ent** $a : Matriu$) **retorna** $real$

{Prec: $FilesMatriu(a) = ColsMatriu(a)$}

{Post: $Determinant(a) =\mid a \mid$}

NOTA: No està permés en aquest problema fer assignacions directes entre una variable de tipus $Matriu$ i una variable o expressió de tipus $Matriu$.

4.11 SolTriangular, exercici 5 de l'examen final del 2/01/2007

Sigui M una matriu quadrada triangular inferior i de dimensió n, amb determinant no nul. Sigui v un vector de dimensió n. Es tracta de fer un subprograma per resoldre el sistema $Mx = v$.

Recordeu que en el cas que la matriu M sigui triangular inferior, la solució ens ve donada per

$x_i = (v_i - \sum_{j=0,\ldots,i-1}(m_{ij} * x_j))/m_{ii}$ per $i = 0, \ldots, n-1$

5 Problemes d'apuntadors

5.1 Desplasfiles, exercici 6 de l'examen final del 1/01/2001

Exercici 2: Desplaçament de files en una matriu (3 punts)

Sigui la implementació del tipus Matriu:

```
typedef struct {
    float **telem; /* Taula d'apuntadors de taules d'elements */
    int nf;        /* Nombre de files */
```

```
    int nc;         /* Nombre de columnes */
} Matriu;
```

1. Dissenyeu un subprograma en C que apliqui una rotació circular d'una fila cap amunt, a totes les files d'una variable de tipus Matriu, usant la implementació donada i de la manera més eficient possible.

2. Dissenyeu un subprograma en C que apliqui una rotació circular de n files cap amunt, a totes les files d'una variable de tipus Matriu, cridant al subprograma dissenyat a l'anterior apartat.

Exemple per $n = 2$:

	1	2	3	4			9	10	11	12
	5	6	7	8			13	14	15	16
La matriu	9	10	11	12		s'ha de convertir en	17	18	19	20
	13	14	15	16			1	2	3	4
	17	18	19	20			5	6	7	8

5.2 ComplexitatSM, exercici 6 de l'examen final del 1/06/2001

Ens proporcionen la següent funció que a partir d'una matriu A, obté la submatriu entre les files $f1$ i $f2$ i les columnes $c1$ i $c2$:

{Prec: $0 \leq f1 \leq f2 < \mathit{FilesMatriu}(A) \ \wedge \ 0 \leq c1 \leq c2 < \mathit{ColsMatriu}(A)$}
funció *ObteSubMatriu*(**ent** $A : Matriu$, **ent** $f1, c1, f2, c2$: **enter**) **retorna** $Matriu$
  var $B : Matriu$; i, j : **enter**; **fvar**
  $B :=$ *CreaMatriu*$(f2 - f1 + 1, c2 - c1 + 1)$
  $i := f1$
  mentre $i <= f2$ **fer**
    $j := c1$
    mentre $j <= c2$ **fer**
      AssigMatriu$(B, i - f1, j - c1,$ *ConsMatriu*$(A, i, j))$
      $j := j + 1$
    fmentre
    $i := i + 1$
  fmentre
  retorna B
ffunció
{Post: *Retorna* B *t.q* *FilesMatriu*$(B) = f2 - f1 + 1 \ \wedge$ *ColsMatriu*$(B) = c2 - c1 + 1 \ \wedge$
  $\forall i, j : f1 \leq i \leq f2, c1 \leq j \leq c2 :$ *ConsMatriu*$(A, i, j) =$ *ConsMatriu*$(B, i - f1, j - c1)$}

a) Quines són les magnituds que determinen la complexitat de l'algorisme?

b) Determineu la complexitat asimptòtica en el cas pitjor de l'algorisme en funció d'aquestes magnituds, suposant que el cost de les operacions bàsiques sobre el tipus *Matriu* és constant.

5.3 ApuntadorsMatriuSimetrica, exercici 6 de l'examen final del 2/01/2004

Definir un tipus en C que utilitza memòria dinàmica per guardar una matriu quadrada simètrica amb el màxim estalvi de memòria possible.

Si pel canal estàndard d'entrada ens donen un enter n, i una matriu simètrica $n \times n$ per files, fer un tros de codi C per guardar aquesta matriu en una variable del tipus definit en l'apartat anterior.

5.4 Apuntadors, exercici 6 de l'examen final del 2/06/2004

Considerem la implementació del tipus Matriu que utilitza memòria dinàmica següent:

```
typedef struct {
    float **telem; /* Taula d'apuntadors de taules d'elements */
    int nf;        /* Nombre de files */
    int nc;        /* Nombre de columnes */
} Matriu;
```

Escriviu un tros de programa en C que a partir de dues variables del tipus `Matriu`, A i B, que tenen el mateix nombre de columnes i el mateix nombre de files, i que ja han estat inicialitzades, construeixi una matriu C amb el mateix nombre de columnes que A i B i amb les files de A en les posicions parells $(0, 2, \ldots)$ i les files de B en les posicions senars. A més les files de A estaran col·locades amb el mateix ordre que en la matriu original i, el mateix passarà amb les files de B. Després de l'operació, A i B seguiran existint. S'han de minimitzar el nombre d'operacions realitzades (reserva i alliberament de memòria, assignacions, etc.). Per exemple:

$$\overbrace{\begin{matrix}0\\1\\2\end{matrix}\begin{pmatrix}1.0 & 2.0 & 3.0\\4.0 & 5.0 & 6.0\\7.0 & 8.0 & 9.0\end{pmatrix}}^{A}, \overbrace{\begin{pmatrix}10.0 & 11.0 & 12.0\\13.0 & 14.0 & 15\\16.0 & 17.0 & 18.0\end{pmatrix}}^{B} \xrightarrow{\text{entrellaça}} \overbrace{\begin{matrix}0\\1\\2\\3\\4\\5\end{matrix}\begin{pmatrix}1.0 & 2.0 & 3.0\\10.0 & 11.0 & 12.0\\4.0 & 5.0 & 6.0\\13.0 & 14.0 & 15.0\\7.0 & 8.0 & 9.0\\16.0 & 17.0 & 18.0\end{pmatrix}}^{C}$$

5.5 ApuntadorsParellsSenars, exercici 6 de l'examen final del 1/01/2005

Ens demanen un programa en C tal que donats pel CEE un enter parell n i els coeficients d'una matriu A $n \times n$ donats per files, construeixi i guardi dues submatrius d'A: la submatriu (A_p) formada pels coeficients que estan en una fila parell i un columna parell i la submatriu (A_s) formada pels coeficients de posicions fila senar i columna senar.
Per exemple,

A partir de $\overbrace{\begin{pmatrix}00.0 & 01.0 & 02.0 & 03.0 & 04.0 & 5.0\\10.0 & 11.0 & 12.0 & 13.0 & 14.0 & 15.0\\20.0 & 21.0 & 22.0 & 23.0 & 24.0 & 25.0\\30.0 & 31.0 & 32.0 & 33.0 & 34.0 & 35.0\\40.0 & 41.0 & 42.0 & 43.0 & 44.0 & 45.0\\50.0 & 51.0 & 52.0 & 53.0 & 54.0 & 55.0\end{pmatrix}}^{A}$ es tindrà $\overbrace{\begin{pmatrix}11.0 & 13.0 & 15.0\\31.0 & 33.0 & 35.0\\51.0 & 53.0 & 55.0\end{pmatrix}}^{A_s}$ i $\overbrace{\begin{pmatrix}00.0 & 02.0 & 04.0\\20.0 & 22.0 & 24.0\\40.0 & 42.0 & 44.0\end{pmatrix}}^{A_p}$

Un equip de programadors ja ha decidit la implementació a fer. Hi hauran dos representacions de tipus:

MatriuQ per les matrius quadrades:

```
typedef struct {
  float **telem;/*Taula de taules de reals*/
  int n;        /*No. de files o columnes */
} MatriuQ;
```

MatriuQPS per una taula de matrius quadrades:

```
typedef struct {
  MatriuQ *m;/*Taula de matrius quadrades */
  int n;     /*Nombre de matrius quadrades*/
} MatriusQPS;
```

... I ja s'ha fet el següent tros de programa:

```
......
  MatriusQPS Ms;
  int    i,j, n;
  float r;

  LlegirEnter(&n);
  Ms = CreaMatriusQPS(2, n / 2);
  i = 0;
  while (i< n) {
    j = 0;
    while (j< n) {
      LlegirReal(&r);
      AssigMatriusQPS(&Ms, i, j, r);
      j = j + 1;
    }
    i = i +1;
  }
.......
```

L'analista en cap ens demana que implementem en C els subprogrames:
`MatriuQ CreaMatriuQ(int n)`
`/* Pre: CERT */`
`/* Post: crea una matriu` $n \times n$ `*/`,
`MatriusQPS CreaMatriusQPS(int n, int dmq)`
`/* Pre: CERT */`
`/* Post: creada una taula de` n `matrius` $dmq \times dmq$ `*/`,
`void AssigMatriusQPS(MatriusQPS const * Matrius, int i, int j, float r)`
`/* Pre: Matrius conté dos matrius quadrades,` r `és el valor` a_{ij} `d'una matriu` A.
`*/`
`/* Post; Si` i `i` j `són parells,` r `es posarà a la matriu 1 (`A_p`) de` $Matrius$ `en la posició corresponent, si` i `i` j `són senars,` r `es posarà a la matriu 0 (`A_s`) de` $Matrius$ `en la posició corresponent. En el cas en que les paritats d'`i `i` j `siguin diferents, no es farà res. */`

5.6 ApPolinomis, exercici 6 de l'examen final del 1/06/2005

Donada aquesta definició de tipus per l'estructura de dades Polinomi:

```
typedef struct{
        int grau;    /* grau màxim */
        float *coef; /* Es guarden tots els coeficients del polinomi */
}Polinomi;
```

Implementeu les operacions:

```
Polinomi CrearPolinomi(int grau)
/*  {Pre: grau = G}
*/
/*  {Post: Creat el polinomi nul de grau màxim G 0 · x^G + 0 · x^(G−1) + ... + 0 · x + 0}
*/
void AssignaCoef(Polinomi *const p,int exp, float coef)
/*  {Pre: p = A_0 + A_1 x + A_2 x^2 + ... + A_n x^n i 0 ≤ exp ≤ n i exp = E i coef = C}          */
/*  {Post: p = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n i a_E = C i per tot i ≠ E, 0 ≤ i ≤ n es té a_i = A_i}  */
void DestruirPolinomi(Polinomi *const p)
/*  {Pre: cert}                                                                              */
/*  {Post: Destruit el polinomi p}
*/
```

Usant la implementació anterior i tenint em compte la següent definició d'estructura de dades que representa un sistema de polinomis,

```
typedef struct{
        int n;       /* nombre de polinomis */
        Polinomi *p; /* taula de polinomis   */
}SistemaPolinomi;
```

implementeu les operacions:

```
SistemaPolinomi CrearSistemaPolinomi(int num, int g)
/*  {Pre: num = N i g = G}                                                                   */
/*  {Post: Creat un sistema de N polinomis nuls de grau màxim G}                             */
void AssignaCoefPolSistemaPolinomi(SistemaPolinomi *const s,int k, int exp, float coef)
/*  {Pre: s = P_0, P_1, P_2, ... P_n i 0 ≤ k ≤ n i k = K i exp = E i coef = C i P_k = A_0 + A_1 x + ... + A_g x^g
    on g és el grau màxim del polinomi}                                                      */
/*  {Post: s = p_0, p_1, ... p_n i per tot i ≠ K, 0 ≤ i ≤ n es té p_i = P_i i P_K = a_0 + a_1 x + a_2 x^2 + ... + a_g x^g
    i a_E = C i per tot i ≠ E, 0 ≤ i ≤ n es té a_i = A_i}                                     */
void DestruirSistemaPolinomi(SistemaPolinomi *const s)
/*  {Pre: cert}                                                                              */
/*  {Post: Destruit el sistema de polinomis s}
*/
```

5.7 ApPoligons, exercici 6 de l'examen final del 1/01/2006

En una aplicació gràfica de dibuix de polígons es té la següent definició pel tipus *Poligon*:

```
typedef struct{
  float x;
  float y;
}Vertex;

typedef struct{
        int nVertexs;    /* nombre de vèrtexs */
        Vertex *vPol; /* taula de vèrtexs */
}Poligon;
```

Implementeu les operacions:

```
Poligon CrearPoligon(int nVertexs)
/*  {Pre: nVertexs = N}
*/
/*  {Post: Creat el poligon nul de N vèrtexs. Tots els seus vèrtex són (0,0)}
*/
void AssignaVertex(Poligon  * const p , int i, Vertex v)
/*   {Pre: p = (X_0,Y_0)(X_1,Y_1)...(X_{n-1},Y_{n-1}) i 0 ≤ i < n i v = (X,Y) i i = I}   */
/*   {Post: p = (x_0,y_0)(x_1,y_1)...(x_{n-1},y_{n-1}) i (x_I,y_I) = (X,Y) i per tot j ≠ I, 0 ≤ j < n es té
     (x_j,y_j) = (X_j,Y_j)}                                                              */
void DestruirPoligon(Poligon *const p)
/*   {Pre: cert}                                                                        */
/*   {Post: Destruit el poligon p}
*/
```

Usant la implementació anterior i tenint em compte la següent definició d'estructura de dades que representa un dibuix,

```
typedef struct{
        int n;       /* nombre de poligons */
        Poligon *p; /* taula de poligons   */
} Dibuix;
```

implementeu les operacions:

```
Dibuix CrearDibuix(int num, int nVertexs)
/*  {Pre: num = N i nVertexs = NV}                                                      */
/*  {Post: Creat un dibuix de N poligons nuls de NV vèrtexs}                            */
void AssignaVertexPolDibuix(Dibuix * const d, int i, int j, Vertex v)
/*   {Pre: s = P_0,P_1,P_2,...P_{n-1} i 0 ≤ k < n i i = I i j = J i v = V i P_k és un polígon}   */
/*   {Post: s = p_0,p_1,...p_{n-1} i per tot k ≠ i, 0 ≤ k < n es té p_k = P_k i p_I és el resultat d'aplicar
     AssignaVertex(P_I,J,V)}                                                             */
void DestruirDibuix(Dibuix *const d)
/*   {Pre: cert}                                                                        */
/*   {Post: Destruit el dibuix d}
*/
```

5.8 Teatre, exercici 6 de l'examen final del 1/06/2006

Per tal de conèixer quins són els seients ocupats d'un teatre, ens proposen la definició del tipus `Teatre` següent:

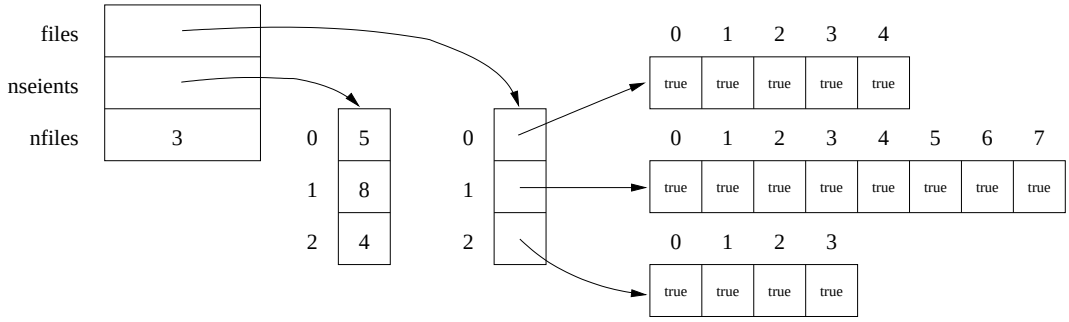
```
typedef struct {
  bool **files;
  int *nseients;
  int nfiles;
} Teatre;
```

Aquesta definició de tipus serveix per teatres on el número de seients pot ser diferent per cada fila. Així, el camp `nseients` és una taula d'enters en la que a la posició `i-1` hi ha el número de seients de la fila `i`. Les files es numeren a partir d'1. El camp `files` és una mena de matriu de booleans amb la particularitat que cada fila pot tenir un número de seients diferent. Així, el booleà corresponent a la posició `i-1`, `j-1` de la matriu indica que el seient

`j` de la fila `i` està lliure si val cert i ocupat si val fals. Els seients es numeren a partir d'1. Es demana que dissenyeu el subprograma

```
Teatre CrearTeatre(void);
```

que llegeix del Canal Estàndard d'Entrada el número de files del teatre i la seqüència de número de seients de cada fila, reserva l'espai necessari i inicialitza tots els seients a lliure. Per exemple, si el CSE conté `3 5 8 4`, `CrearTeatre` retorna un objecte de tipus Teatre com el següent:



Per tal de llegir enters, podeu utilitzar el subprograma

```
void LlegirEnter(int *const a);
```

5.9 MatriuEnterGrans, exercici 6 de l'examen final del 2/01/2007

Es vol fer una calculadora de matrius de màxima precisió. Per aquest motiu s'ha decidit que els elements de la matriu siguin del tipus *enterGran* que definim a continuació com:

```
typedef struct {
  bool negatiu;
  int *x; /* taula de Xifres */
} enterGran

typedef struct {
    enterGran **telem; /* Taula d'apuntadors de taules d'elements */
    int nf;        /* Nombre de files */
    int nc;        /* Nombre de columnes */
    int nx;        /* Nombre de xifres dels enters grans*/
} matriuEG;
```

Es demana implementeu les següents funcions i/o accions:

```
matriuEG crearMatriuEG(int nf, int nc, int nx)
 /* Pre: nf > 0 i nc > 0 i nx > 0*/
 /* Post: Creada una matriu de nf files, nc columnes d'enters de nx xifres que valen 0.*/

void assigDigitEnterMatriuEG(matriuEG * const m, int i, int j, int pes, int digit)
/* Pre: 0 ≤ i < (∗m).nf ∧ 0 ≤ j < (∗m).nc ∧ 0 ≤ pes < (∗m).nx ∧ 0 ≤ digit < 9
 i l'enter que està a la posició i,j de m val E i E = ∑_{i=0}^{(∗m).nx−1} (e_i · 10^i) */
/* Post: l'enter que està a la posició i,j de m val
          ∑_{i=0}^{pes−1} (e_i · 10^i) + digit · 10^pes + ∑_{i=pes+1}^{(∗m).nx−1} (e_i · 10^i) */
```

Índex