Poseu el nom a tots els fulls
Entregueu els problemes en fulls separats
Les respostes han de ser justificades

Problema 1 *3 punts* Temps estimat: 30 min

Un amic vostre és un gran amant del *Tuning*. Després d'haver fet moltes modificacions en el seu cotxe decideix canviar el xip del injector. Després d'avaluar els diferents models existents, i veure que cap d'ells dona un bon rendiment, pren la decisió de programar ell mateix el xip. Per fer-ho us demana ajuda a vosaltres.

L'algorisme a dissenyar haurà de llegir les dades d'un fitxer d'entrada 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 haurà de crear un fitxer de sortida "cabals.dat" en el qual hi apareixerà la seqüència de cabals que s'han de disparar en funció de les revolucions per minut reals del motor. 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 i la velocitat del motor indicar quin cabal aplicar. La primera funció s'aplica sempre que la temperatura del motor és menor que 100 graus i la velocitat del cotxe és menor que 200 Km/h. En cas contrari s'aplicarà la segona funció.

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 $encurs(f) = \beta$ i $pd(f) = \gamma\delta$ i $pd(f) \neq$ 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*(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*(rpm : **real**, t : **real**, 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 }

Problema 2 *4 punts* Temps estimat: 30 min

Una empresa de serveis d'internet té dos servidors de pàgines. Cada servidor genera automàticament un fitxer FST que conté els noms de les pàgines que han sigut visitades amb el seu nombre de visites. Les pàgines en cada FST 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 han 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:

acció *llegirPaginaFST*(**entsor** $f : FST$, **sor** $nom : tMot$, **sor** $visites$: **enter**)
{**Pre:** f es un FST 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

Problema 3 *3 punts* Temps estimat: 30 min

Ens proporcionen els següent algorisme:

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

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

Es demana:

a) Calcular la complexitat assimptòtica en el cas millor i també en el cas pitjor d'aquest algorisme. Identifiqueu els casos millor i pitjor.

b)Es pot millorar la complexitat l'algorisme, sense variar l'especificació, a fi que la seva complexitat es redueixi?.

c)Si es pot canviar l'especificació per millorar la seva complexitat, escriviu la nova especificació i indiqueu en quin algorisme conegut es basaria la nova solució indicant també la nova complexitat resultant.