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

Problema 1 *4 punts* **Temps estimat: 60 min**

Dos nens que fan una col·lecció de cromos volen fer un intercanvi dels cromos segons el sistema de tota la vida d'intercanvi d'un repetit (repe) per un que els manca (falti). Els cromos estan identificats per un enter entre 1 i 200, ja que la col·lecció completa és de 200 cromos.

Disposem del tipus *ColeCromos* que permet representar el conjunt de cromos de la col·lecció que té un nen, amb les següents operacions:

acció *Inicialitzar*(**sor** c : *ColeCromos*)
{**Pre: cert**}
{**Post:** c és una col·lecció buida de cromos, és a dir, per tot i, $1 \leq i \leq 200$, $Freq(c, i) = 0$}

acció *AfegirCromo*(**entsor** c : *ColeCromos*, **ent** id : **enter**)
{**Pre:** $Freq(c, id) = N \ \wedge \ N \geq 0 \ \wedge \ 1 \leq id \leq 200$ }
{**Post:** $Freq(c, id) = N + 1$}

acció *TreureCromo*(**entsor** c : *ColeCromos*, **ent** id : **enter**)
{**Pre:** $Freq(c, id) = N \ \wedge \ N > 0 \ \wedge \ 1 \leq id \leq 200$ }
{**Post:** $Freq(c, id) = N - 1$}

funció *Freq*(**ent** c : *ColeCromos*, **ent** id : **enter**)
{**Pre: cert**}
{**Post:** $Freq(c, id)$ és la freqüència del cromo id en c, és a dir, el nombre de vegades que s'ha afegit el cromo id menys el nombre de vegades que s'ha tret}

Fixem-nos que aquest tipus és una ampliació de les taules de freqüències; disposa d'una operació addicional que permet descomptar una aparició d'un cromo. El significat del valor f que retorna la crida a $Freq(c, x)$ és el següent:

$f = 0$: el cromo x no és a la col·lecció c (falti);
$f = 1$: el cromo ja el tenim a la col·lecció (tengui), però no en disposem de cap repetit;
$f > 1$: el cromo el tenim a la col·lecció i a més a més el tenim repetit $f - 1$ vegades (repe).

Es demana que dissenyeu una acció que a partir de dues *ColeCromos* que representen el conjunt de cromos de què disposen cadascun dels dos nens, les actualitzi fent tots els intercanvis de cromos, començant pels cromos que tenen el nombre més baix. Dissenyeu en primer lloc una acció que utilitzi el següent subprograma

funció *seguentIntercanviable*(**ent** $c1, c2$: *ColeCromos*, **ent** id : **enter**) **retorna enter**
{**Pre:** id=0 o id és el darrer cromo que ha donat la col·lecció 1 a la col·lecció 2}
{**Post:** retorna, en el cas que existeixi, el següent cromo després d'id que hi ha repe a $c1$ i manca a $c2$. En cas de no ser així, retorna un nombre més gran que 200}

En segon lloc, dissenyeu el subprograma *seguentIntercanviable*.

Per exemple, si inicialment les col·leccions de cromos dels dos nens són aquestes (els no escrits és que no els tenen):

NEN1		NEN2	
Cromo	Freq	Cromo	Freq
1	2	1	1
2	3	3	2
4	1	6	10
5	5	7	3
8	3	8	4
9	2	11	1
10	5		
14	2		

Després de fer la crida seran aquestes:

NEN1		NEN2	
Cromo	Freq	Cromo	Freq
1	2	1	1
2	2	2	1
3	1	3	1
4	1	5	1
5	4	6	9
6	1	7	2
7	1	8	4
8	3	9	1
9	1	11	1
10	5		
14	2		

S'hauran intercanviat 3 cromos: el NEN1 haurà obtingut els cromos 3, 6 i 7 i l'hi haurà donat al NEN2 els cromos 2,5 i 9.

PROBLEMA 2 i 3 AL DORS

Problema 2 ***3 punts*** **Temps estimat: 45 min**

Donades dues matrius de $n \times n$,

$$A = \begin{pmatrix} a_{0,0} & \dots & a_{0,n-1} \\ \vdots & \dots & \vdots \\ a_{n-1,0} & \dots & a_{n-1,n-1} \end{pmatrix} \text{ i } B = \begin{pmatrix} b_{0,0} & \dots & b_{0,n-1} \\ \vdots & \dots & \vdots \\ b_{n-1,0} & \dots & b_{n-1,n-1} \end{pmatrix}$$

dissenyeu un subprograma que retorni cert si A i B satisfan:
$\sum_{k=0}^{n-1}(a_{ik}^2 + a_{kj}^2) = \sum_{k=0}^{n-1}(b_{ik}^2 + b_{kj}^2)$, per tot $i, j : 0 \leq i, j \leq n-1$

Problema 3 ***3 punts*** **Temps estimat: 30 min**

Es vol mantenir un diccionari de sinònims amb el mínim de memòria possible. Per fer-ho es disposa del tipus

```
typedef struct {
  Entrada * e;      /* taula d'entrades*/
  int nEntrades;  /* Nombre d'entrades del diccionari */
} Diccionari;


typedef struct {
  Paraula p;             /* paraula a la que s'associen els sinònims */
  Paraula *s;             /* taula de sinònims  */
  int     nSinomims;    /* nombre de sinònims */
} Entrada;


typedef struct {
  char *c;  /* taula de caràcters  */
  int  nCar;   /* nombre de caràcters */
} Paraula;
```

Implementeu les següents accions:

```
Paraula creaParaula(int n, char * taulaCar);
/* Pre: n=N, en taulaCar hi ha  N caràcters de la paraula a crear */
/* Post: retorna una paraula amb els N caràcters de taulaCar */

Entrada creaEntrada(int n, char * taulaCar, int m, Paraula * s);
/* Pre: n=N, en taulaCar es troben N caràcters de la paraula que forma
   l'entrada, i m=M és el nombre de sinònims de la taula de paraules que apunta s */
/* Post: retorna una entrada per la paraula de N caràcters que es troba a
   taulaCar, junt amb els seus M sinònims trobats on apunta  s*/

Diccionari creaDiccionari(int n);
/*Pre: n=N i N>0*/
/*Post: retorna un diccionari amb espai reservat per N entrades */
```