Part 3

- Ara veurem com usar memòria dinàmica si volem treballar amb matrius.

1 Volem calcular el producte d'una matriu de dimensió $n \times m$ per un vector de dimensió m. Si $A = (a_{ij}), 1 \leq i \leq n, 1 \leq j \leq m$ i $u = (u_i), 1 \leq i \leq m$, llavors el producte, $v = Au$, ve donat per

$$v_i = \sum_{j=1}^{m} a_{ij}u_j = a_{i1}u_1 + \cdots + a_{im}u_m, i \in \{1, \ldots, n\}.$$

El següent programa llegeix les dimensions de la matriu, les components de la matriu i del vector i calcula el seu producte.

```
/*C\'alcul del producte d'una matriu per un vector usant mem\'oria din\'amica*/
#include<stdio.h>
#include<stdlib.h>

int main(void){
  int n, m, i, j;
  float **a, *u, *v;
  printf("Doneu les dimensions de la matriu, (n,m) = \n");
  scanf("%d %d", &n,&m);

  a = (float **)malloc( n*sizeof(float *) );
  if ( a == NULL ){
     printf("No hi ha prou mem\'oria");
     exit(1);
  }

  for (i = 0; i < n; i++){
      a[i] = (float *)malloc( m*sizeof(float) );
      if ( a[i] == NULL ){
           printf("No hi ha prou mem\'oria");
           exit(2);
      }
  }

  u = (float *)malloc( m*sizeof(float) );
  v = (float *)malloc( n*sizeof(float) );
  if ( u == NULL || v == NULL ){
     printf("No hi ha prou mem\'oria");
     exit(3);
  }

  printf("Doneu els (%d x %d) elements de la matriu A \n", n, m);
  for (i = 0; i < n; i++){
      for (j = 0; j < m; j++){
          scanf("%f", &a[i][j]);
      }
  }

  printf("Doneu els %d elements del vector u \n", m);
```

```
  for (i = 0; i < m; i++)
     scanf("%f", &u[i]);

  for (i = 0; i < n; i++){
      v[i] = 0.f;
      for (j = 0; j < m; j++){
          v[i] += a[i][j]*u[j];
      }
  }

  printf("El␣producte␣de␣la␣matriu␣A␣=␣\n");

  for (i = 0; i < n; i++){
      for (j = 0; j < m; j++){
          printf("␣%16.7e␣", a[i][j]);
      }
      printf("\n");
  }

  printf("pel␣vector␣u␣=␣\n");

  for (i = 0; i < m; i++)
      printf("␣%16.7e␣\n", u[i]);

  printf("ens␣d\'ona␣v␣=␣\n");

  for (i = 0; i < n; i++)
      printf("␣%16.7e␣\n", v[i]);

  for (i = 0; i < n; i++)
      free (a[i]);

  free(a);
  free(u);
  free(v);

  return 0;
}
```

a) Executeu el programa per a $n = 2$, $m = 3$ i

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \quad u = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

b) Executeu-lo per a diferents valors de n, m, A i u.

- A continuació cal escriure un programa que recull el que hem vist: llegir dades d'un fitxer, usar assignació dinàmica de memòria per a vectors i matrius, comparar dades, usar funcions guardades en diferents fitxers...

2 Feu una funció de nom `prod`, que calculi el producte d'una matriu per un vector.

Feu una funció principal, que usant la funció `prod`, calculi el vector $y = ABx$ i determini si y és múltiple de x. No calculeu el producte AB.

S'han de llegir dos enters, m i n, una matriu real A, emmagatzemada per files, de dimensió $m \times n$, una matriu real B, emmagatzemada per files, de dimensió $n \times m$, i un vector x de m components reals.

Caldrà escriure A, B i x en forma de matriu, calcular el vector $y = ABx$ i escriure'l, obtenir els mòduls de x i y i escriure'ls i, finalment, determinar si y és múltiple de x, donant el missatge corresponent.

- Exercici d'autoavaluació:

3 Escriviu una funció que, donades les matrius A $(n \times p)$ i B $(p \times m)$ i n, p i m dimensions, retorni la matriu $C = AB$.

Recordeu que

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj},\ 1 \leq i \leq n,\ \ 1 \leq j \leq m$$

Useu aquesta funció per a calcular AB i dir si és possible calcular BA. Aplicar aquesta funció per calcular AB i $B^t A^t$ per les matrius

$$A = \begin{pmatrix} 0.1 & 1.2 \\ 0.4 & -1.1 \\ 0.03 & 20.1 \end{pmatrix} \quad B = \begin{pmatrix} 0.11 & -10.1 & 1.23 \\ 9.89 & -7.99 & 0.96 \end{pmatrix}$$

La funció `main` ha d'usar memòria dinàmica i llegirà les dimensions de A i les seves components per files i després les dimensions de B i les seves components també per files.