Exercice 1 : Pointeurs et Adresses

- Donner la trace de l'exécution de ce programme
- Saisir et exécuter ce programme
- Analyser les résultats issus de l'exécution

```
#include<include.h>
main() {
float *px ; /*déclaration d'un pointeur sur un réel */
float x = 45.9 ;
px = &x ; /* px pointe sur x */
printf(''Adresse de x : %x \n'', &x) ;
printf(''Valeur de px : %x \n'', px) ;
printf(''Valeur de x : %4.1f \n'', x) ;
printf(''Valeur pointée par px : %5.1f \n'', *px) ;
return 0;
}
```

Exercice 2 : Considérons les fonctions suivantes :

passage de paramètres par valeurs

```
void permut(int a, int b) {
int c ;
c = a ; a = b ; b = c ;
return ; }
```

passage de paramètres par adresse

```
void permut (int *a, int  *b) {
int c ;
c = *a ; * a = *b ; *b = c;
return ; }
```

- Expliquer la différence entre ces deux fonctions.
- Ecrire un programme qui permet de saisir au clavier deux entiers naturels, de définir et d'appeler ces deux fonctions en affichant, avant et après leurs appels, les valeurs des variables passées en arguments à ces deux fonctions.
- Analyser les résultats issus de l'exécution.

Exercice 3 : Allocation dynamique de l'espace mémoire

- Ecrire une fonction qui permet de saisir les valeurs d'une matrice réelle.
- Ecrire une fonction qui permet d'afficher les valeurs d'une matrice réelle.
- Ecrire une fonction qui permet d'allouer d'une façon dynamique de l'espace mémoire pour une matrice réelle.
- Ecrire une fonction qui permet de faire le produit de deux matrices réelles de types $A(m,n)$ et $B(n,p)$. Le résultat sera dans la matrice $AB(m,p)$.
- Ecrire une fonction qui permet de libérer l'espace occupé par une matrice réelle.
- Ecrire un programme qui utilise les fonctions, ci-dessus, pour effectuer le produit de deux matrices réelles.

-Fin-