INFO505 – TD2 – 2024-2025

Exercice I : Echauffement

```
int main() {
  int i=1;
  int *ptr = &i;
  printf("%x\n", ptr);
  printf("%x\n", *ptr);
  printf("%x\n", &(*ptr));
  printf("%x\n", *(&ptr));
  return 0;
}
```

```
#include <stdio.h>

int main() {
  int a, b;
  int *ptr1, *ptr2;
  a = 5;
  b = a;
  ptr1 = &a;
  ptr2 = ptr1;
  b = (*ptr2)++;
  printf("a = %d, b = %d, *ptr1 = %d, *ptr2 = %d\n",
a, b, *ptr1, *ptr2);
  return 0;
}
```

```
void main (void)
{
  int *p;
  int a = 0;
  *p = 4;
…printf("%d",p) ;
}
```

```
#include <stdio.h>

void trio(int a, int b, int c) {
  a = b + c;
  b = c + a;
  c = a + b;
}
int main(void) {
  int a = 1, b = 2, c = 3;
  trio(a, b, c);
  printf("%d %d %d\n", a, b, c);
  return 0;
}
```

```
#include <stdio.h>
#define TAB_LENGTH 3

int main() {
  int tab[TAB_LENGTH];
  int j = 0;
  int *ptr = tab;
  for(; j < TAB_LENGTH; j++)
     tab[j] = 5;
  *(ptr + 1) = 3;
  printf("[ %d %d %d ]\n", tab[0], tab[1], tab[2]);
  return 0;
}
```

```
  int a[]={1,2,3,4,5};
  int *ptr  = a;
  printf("%d %d %d \n",a[1],a[3], a[a[3]]) ;
  printf("%d \n",*(ptr+1)) ;
```

```
  int nombre = 8 ;
  int *p_int = &nombre ;
  printf ("%p \n", p_int);
```

On suppose que la variable *nombre* est stockée à l'adresse 0x5000 et *p_int* à l'adresse 0x2300.

Exercice II : Pile d'exécution

```
void swap ( int * p , int * q )
{
        int t = *p ;
        *p = *q ;
        *q = t ;
}
void mystere ( int * f , int * m, int * l )
{
        int * n = m;
        // (2) s i t u a t i o n i n i t i a l e
        while ( f != n )
        {
                swap ( f++, n++ ) ;
                if ( n == l ) n = m;
                else if ( f == m ) m = n ;
        }
        // (3) s i t u a t i o n en f i n de bouc l e
}
```

```
int main ( )
{
        int t [ 5 ] = { 1 , 0 , 7 , 4 , 3 } ;
        // (1)
        mystere ( t , t + 2 , t + 5 ) ;
        // (4)
        for ( int i = 0 ; i < 5 ; ++i )
        {
                p r i n t f ( " %d" , t [ i ] ) ;
                p r i n t f ( "\n" ) ;
        }
return 0 ;
}
```

1. Afficher la pile d'exécution de ce programme aux lignes (1), (2), (3) et (4), comme précisé dans le code.
2. Que contient le tableau t à la fin ? Plus généralement, que fait la fonction mystere() ?

Exercice III : Fonctions polynomiales

Une fonction polynomiale de degré n peut être représentée par un tableau de n + 1 coefficients.

```
void p_sum (float * R, const float * P1 , const float * P2 , unsigned int degree) ;

int main () {
        // P1 : x^2 + 2x + 3
        // P2 : 3x^3 + x^2 + 4
        // P3 : 3x^3 + 2x^2 + 2x + 7 (c'est-à-dire P1+P2)
        float P1 [4] = { 3., 2., 1., 0. } ;
        float P2 [4] = { 4., 0., 1., 3. } ;
        float P3 [4] ;
        p_sum (P3 , P1 , P2 , 3) ;
        for ( int i = 0 ; i <= 3 ; ++i) {
                printf (« c_%d␣=␣%.1f\n », i, P3[i]) ;
        }
        return 0 ;
}
```

1. Qu'est-ce qui ne va pas dans ce programme ? À quelle étape la « compilation » échoue-t-elle ?
2. Corrigez cette erreur et testez le programme.
3. Implémentez une fonction qui évalue une fonction polynomiale P au point x[1].

```
float p_eval (float x, const float * P, unsigned int degree) ;
```

[1] *Optionnellement, vous pouvez implémenter la méthode efficace du schéma de Horner.*