TP N°1 : programmation C++

Exercice 1

Question 1 :

```
int A = 1, B = 2, C = 3;
int *P1, *P2;

P1 = &A;
P2 = &C;
*P1 =(*P2)++;  // A=C; C=C+1; (A=3; C=4)
P1=P2;
P2 = &B
*P1 -=*P2;    // C = C - B; (C=2; B=2)
++*P2;       // B = B+1; (B=3)
*P1 *= *P2; // C = C * B; (C=6; B=3)
A = ++*P2**P1; //A=(++B)*C (A=24; B=4; C=6)
P1 = &A;
*P2 = *P1/=*P2; // A=A/B; B=A (A=6; B=6)
```

Après l'exécution de ce programme :
A = 6
B = 6
C = 6
P1 = &A
P2 = &B

Question 2 : Un pointeur et une référence

- Déclarez un tableau d'entiers A de dimension 10

```
int *A = new int[10];
```

ou

```
int A[10] ;
```

- Développez deux méthodes, remplitA et afficherA

```
void remplitA (int* A, int n)
{
      for (int i=0; i<n; i++)
      {
            cin >> A[i];
      }
}
```

```
void afficherA (int* A, int n)
{
      for (int i=0; i<n; i++)
      {
            cout << A[i] << endl;
      }
}
```

- Déclarez une référence vers l'élément 5 du tableau et incrémenter sa valeur

```
int &x = A[4];
x++;
```

- Déclarez un pointeur vers l'élément 7, incrémenter le pointeur et la valeur pointée

```
int *P = A+6 ;
P++ ;
*P++ ;
```

Question 3 : Passage par valeur vs Passage par référence vs Passage par pointeur

```
void square(int n) {
cout << "In square(): " << &n << endl; // 0x22ff00
n *= n;
}
```

```
int main()
{
 int n;
 cin >> n;

 square (n);
 cout << n;//la valeur de n ne change pas

}
```

```
void square(int * pNumber) {
cout << "In square(): "<< pNumber << endl; //0x22ff1c
*pNumber *= *pNumber;
}
```

```
int main()
{
   int n ;
   cin >> n ;

   square (&n) ;
   cout << n;// la valeur de n change
}
```

```
void square(int & rNumber) { )
cout << "In square(): "<< &rNumber <<endl; //0x22ff1c
rNumber *= rNumber
}
```

```
int main()
{
   int n;
   cin >> n;

   square (n);
   cout << n;// la valeur de n change
}
```

On remarque que l'adresse du paramètre rNumber est la même que celle de n

Exercice 2

Trouvez les erreurs dans les suites d'instructions suivantes et proposez les corrections :

a)	b)	c)	d)	e)

`int * p;`
`int x=34;`
`*p=x;`
	`int x=17;`
	`int * p=x;`
	`*p=17 ;`
		`double * q;`
		`int x=17;`
		`int * p=&x;`
		`q=p ;`
			`int x;`
			`int * p;`
			`&x=p;`
				`char mot[10];`
				`char car='A';`
				`char * pc=&car;`
				`mot=pc;`

a. L'erreur : ***p =x ;**

Le pointeur p pointe vers l'inconnu. Pour le faire pointer vers x, on doit écrire **p = &x;**

b. L'erreur : **int* p = x ;**

Pour pointer vers un objet il faut affecter l'adresse de ce dernier

Correction : **int* p = &x ;**

Remarque : après la correction l'affectation *p = 17 serait inutile.

c. L'erreur : **q = p ;**

Chaque pointeur a son type spécifique, donc il ne peut pointer que vers un objet du même type. On pourrait caster **q=(double*)p** pour éliminer l'erreur de compilation mais toute future affectation de la forme (*q = valeur double) causerait des erreurs d'accès de mémoire vu la différence entre la taille d'un double et d'un int).

d. L'erreur : **&x = p ;**

Correction : **p =&x ;**

e. L'erreur : **mot = pc;**

Puisque le tableau est statique, on ne peut pas changer son adresse.

Correction : **mot[0] = *pc;** ou **mot[0] = car ;**

Exercice 3

Ecrire plus simplement en C++, en utilisant les opérateurs *new* et *delete* :

```
int* adi;
double *add;
……
adi = (int*) malloc (sizeof (int));
add = (double*) malloc (sizeof (double) * 100);
```

```
int* adi;
double * add ;
……
adi = new int;
add = new double[100];
```

Exercice 4

Que faut-il modifier pour obtenir une version correcte ?

```
int* mauvais1 (int n)
{
int tab[n];
for (int i=0 ; i<n ; i++))
    cin >> tab[i];
return tab;
}
```

```
Const int MAX = 50 ;
int* mauvais2 ()
{
int tab[MAX];
for (int i=0 ; i<MAX ; i++))
    cin >> tab[i];
return tab;
}
```

Erreurs : On ne peut pas

a. Créer des tableaux statiques à taille variable
b. Retourner un tableau déclaré localement.

Correction : on doit utiliser **la déclaration dynamique**.

```
int* MethodeCorrecte1(int n)
{
  int* tab = new int[n];
  for (int i=0; i<n; i++)
         cin >> tab[i];
    return tab;
}
```

Exercice 5

Ecrire un programme qui déclare un tableau dynamique de chaine de caractères et qui, après avoir demandé le nombre de chaines à stocker, en effectue la saisie.

Vous allez implémentez ces deux cas :

- Le tableau est crée avec un *new*. Les chaines de caractères sont de type `char*`.
- Utiliser la classe vector et la classe string

```
#include <iostream>
using namespace std;

int main()
{
    int n, m;
    cout << "nombre de chaines : ";
    cin >> n;
    cout << "taille des chaines :";
    cin >> m;

    char** M = new char* [n];
    for (int i = 0; i < n; i++)
        M[i] = new char[m+1];

      return 0 ;
}
```

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main()
{
       vector <string> M;
       string s;

       int n;
       cout << "nombre de chaines : ";
       cin >> n;

       for (int i = 0; i < n; i++)
       {
              cin >> s;
              M.push_back(s);
       }

       return 0;
}
```