Langage C : TD4

Exercice 1 : On considère le code suivant :

```
void exercice1()
{
  int i, t[10];
  for(i = 0 ; i < 10 ; i++)
  {
    t[i] = i + 1;
  }
}
```

En supposant que les int sont codés sur 4 octets, et que la valeur $t[0]$ soit stockée à l'adresse 3204, donner les valeurs associées aux expressions suivantes :

1. t : 3204
2. $*t$: 1
3. $\&t[0]$: 3204
4. $\&t[2]$: 3212

Exercice 2 : On considère le code suivant :

```
void exercice2()
{
  int t[] = {7, 16, 23, 38, 41, 52, 71};
  int *p;

  p = t;
}
```

Quelles valeurs ou adresses sont données par les instructions suivantes ?

1. $*(p+2)$: 23 (soit $t[2]$)
2. $p+2$: $\&t[2]$
3. $*p+2$: 9 (soit $t[0]+2$)
4. $t+2$: $\&t[2]$
5. $p+(*(p+4)-*(p+3))$: $\&t[0]+3$ (soit $t[4]-t[3]$)
6. $*(p+t[5]-*(p+4)-*p)$: $t[4]$

Détail du 6. :

$(p+t[5]-*(p+4)-*p) \rightarrow p+t[5]-t[4]-t[0]$

$= p+4 \Rightarrow *(p+4) = t[4]$

Exercice 3 : Que fait le code suivant ?

```
void p(int t[])
{
  int i, s;

  s = sizeof(*t)/sizeof(*t[0]);

  /*Ici sizeof(t) nous donne le nombre d'éléments qu'il peut contenir.
```

```
  sizeof(t[0]) montre la taille en mémoire de l'élément.
  */

  for(i = 0 ; i < s ; i++) printf("%d", t[i]);
}

int main()
{
  int s, t[4] = {1, 2, 3, 4};

  s = sizeof(t)/sizeof(t[0]);
  /*Ici sizeof(t) nous donne la taille en mémoire du tableau.
  sizeof(t[0]) montre la taille en mémoire de l'élément.
  */

  printf("%d\n", s);
  p(t);

  return 0;
}
```

Exercice 4 : Quel sera le résultat du code suivant :

```
void f(double *t, int n)
{
  int i;

  for(i = 0 ; i < n ; i++) *(t + 1) = *(t + i) + 2.0; // t[1] = t[n] + 2
}

int main()
{
  int i, n;
  double t[] = {8.5, 4.0, -1.2, 4.7}, s = 0.;
  //8 octets alloués par double. => sizeof(t) = 32.
  //*t = t[0] => sizeof(t) = 8.

  n = sizeof(t) / sizeof(*t); // = 4
  for(i = 0 ; i < n ; i++) s += *(t+i); //additionne chaque case du tableau ( = 16)


  printf("%f\n", sizeof(t)/s); // 32/16 = 2

  f(t, n); // t[1]= 4.7 + 2.0 = 6.7

  for(i = 0 ; i < n ; i++) printf("%f ", t[i]); // Affiche toutes les cases du tableau
  printf("\n");

  // Bilan : seul t[1] a été impacté.

  return 0;
}
```

Exercice 5 : On donne la fonction main suivante :

```
#include <stdio.h>

#define MAX 4
```

```c
int main()
{
  double t[MAX] = {4.5, 1.0, -2.4, 5.2};
  affiche(t, MAX);
  return 0;
}
```

1. Écrire une version de la fonction affiche pour afficher les n premières valeurs du tableau via un parcours en utilisant un compteur entier et en utilisant la notation [ ] pour accéder aux éléments du tableau.

```c
void affiche(double t[], int n)
{
  int i;

  for(i = 0 ; i < n ; i++) printf("%0.2lf\t", t[i]);
  printf("\n");
}
```

2. Écrire une autre version de la fonction en utilisant un compteur entier mais sans utiliser les [ ] pour accéder aux éléments du tableau.

```c
void affiche(double *t, int n)
{
  int i;

  for(i = 0 ; i < n ; i++) printf("%0.2lf\t", *(t + i));
  printf("\n");
}
```

3. Écrire une autre version dans laquelle on utilise un pointeur pour parcourir les éléments (et donc pas de compteur entier). Dans cet algorithme on affichera les valeurs contenues dans les cases du tableau ainsi que leur adresse.

```c
void affiche(double *t, int n)
{
  double *p = t;
  while (p != t + n)
  {
    printf("%0.2lf ", *p);
    p++;
  }
}
```