TD 10 - Algorithmique et programmation
Révisions

Exercice 1

On souhaite observer les changements de la mémoire suite à certaines instructions.
On supposera les conditions typiques d'adressage connexe et croissante. On commence à 0x100, sizeof(int)=sizeof(void *)=4.

```
1 int a = 1;
2 int b = 2;
3 int* pt=&b;
4 int* pt2=&a;
```

```
1 int * c = (int*)calloc(5,sizeof(int));
2 *c = a;
3 *(c+1) = *pt2;
4 c[1] = 5;
5 c[2] = 10;
```

```
1 float f[3] = {3.5, 2, 10.1}; //sizeof(float) = 8
2 void * pt3 = &f+1;
3 char * mot = "bonjour"
```

Exercice 2

Rediger sur papier un algorithme de tri à bulles en pseudo-code.

Exercice 3

```
1  void plusDeux(int* a) { *a += 2; return; }
2  void moinsUn(int* a) { *a -= 1; return; }
3
4  void plusProcheMult(int* c, int* d)
5  {
6    int manque;
7          int* a = c;
8          int* b = d;
9          int (*pointeur)();
10         if (*a <= *b) { int tmp = a; a = b; b = tmp; }
11         while (*a % *b != 0)
12         {
13                 printf("%d %d\n", *a, *b);
14                 pointeur = plusDeux;
15                 if (*a % *b == 1) { pointeur = moinsUn; manque = manque-3; }
16                 pointeur(a);
17     manque = manque+2;
18         }
19         printf("fini : %d %d\nreste : %d\n", *a, *b, manque);
20 }
21
22 void  main() {
23         int a[2] = { 19,12 };
24         plusProcheMult(a, a + 1);
25 }
```

1. Qu'affiche ce programme ? Que fait-il ?

2. A quoi sert la ligne 12 ?

3. De manière similaire à plusDeux et moinsUn, proposer un fonction en C qui multiplie deux entiers et range le résultat dans l'adresse du premier entier.
   Cette fonction pourra être appellée par pointeur.

4. A l'aide de cette nouvelle fonction,

Exercice 4

Soit l'arbre suivant :

```
1  typedef struct node_ {
2          int value;
3          struct node_* gauche;
4          struct node_* droite;
5  } node;
6  typedef node* pNode;
7
8  pNode newNode(int a)
9  {
10         pNode n = (pNode)malloc(sizeof(struct node_));
11         n->value = a;
12         n->droite = NULL;
13         n->gauche = NULL;
14         return n;}
15
16 //les fonctions addNodeLeft et addNodeRight ajoutent une feuille
17 //respectivement a gauche et a droite d'une racine
18 pNode root = newNode(10);
19 pNode cinq = addNodeLeft(5, root);
20 pNode deux = addNodeLeft(2, deux);
21 pNode vingt = addNodeRight(20, root);
22 pNode quinze = addNodeLeft(15, vingt);
```

1. Dessiner l'arbre sur une feuille.
2. Proposer une algorithme en pseudo-code qui affiche tous les éléments de cet arbre.