Langage C : TD2

Exercice 1 : Quels seraient les résultats / affichages du code suivant ?

```c
#include <stdio.h>

int f(int n)
{
  return n+2;
}

int main()
{
  int a = 5;
  printf("%d %d\n", f(a+2), f(f(a+2)));
}
```

La sortie sera "9 11"

Exercice 2 : Quels seraient les résultats / affichages du code suivant ?

```c
#include <stdio.h>

int f(int a, int b)
{
  a = 2*a+b;
  return a;
}

int main()
{
  int a = 2, b = 5;
  b = f(b, a);
  a = f(b, a);
  printf("%d %d", a, b);
  return 0;
}
```

La sortie sera "26 12"

Exercice 3 : Écrire une fonction récursive qui retourne le chiffre le plus à gauche d'un entier donné.

```c
#include <stdio.h>

int gauche(int n)
{
  if(n/10 == 0) printf("%d", n);
  else gauche(n/10);
}

int main()
{
  gauche(8541);
```

```
  return 0;
}
```

Exercice 4 : Écrire une fonction qui vérifie si un entier naturel donné est un nombre de Armstrong. Un entier n contenant p chiffres est un nombre de Armstrong s'il est égal à la somme des puissances $p - ièmes$ de ses chiffres. Ainsi par exemple $1634 = 1^4 + 6^4 + 3^4 + 4^4$ est un nombre de Armstrong. Compléter ensuite avec un programme qui affiche tous les nombres de Armstrong inférieurs à une valeur lue au clavier.

```
#include <math.h>
#include <stdio.h>

void armstrong(int n)
{
  int p = 1;
  int test = n;

  while(test/10 != 0)
  {
    test /= 10;
    p++;
  }

  int test1 = test = n;
  int somme = 0;

  for(int i=0 ; i<p ; i++)
  {
    test1 /= 10;
    somme += pow((test - test1*10),p);
    test /= 10;
  }

  if(somme == n) printf("\n%d est un nombre d'Armstrong", n);
}

void liste(int n)
{
  for(int i = 1 ; i<n+1 ; i++)
  {
    armstrong(i);
  }
}

int main()
{
  liste(1634);

  return 0;
}
```

Exercice 5 : Écrire une fonction qui calcule et retourne la valeur des n premiers termes de la suite $9 + 99 + 999 + 9999 + ...$, pour une valeur de $n \leq 19$.

```c
#include <stdio.h>

void suite(int n)
{
  if(n<20)
  {
    int somme = 9;
    for(int i=0 ; i<n ; i++)
    {
      printf("%d ", somme);
      somme = somme*10 + 9;
    }
  }

}

int main()
{
  suite(5);

  return 0;
}
```

Exercice 6 : Écrire une fonction récursive qui prend un entier en entrée et retourne cet entier en inversant les chiffres qui le composent. Par exemple la fonction retourne 4321 si le nombre initial est 1234. Ajouter une fonction qui utilise la précédente pour déterminer si un entier donné est un palindrome numérique, puis une fonction main qui teste cette fonction.

```c
#include <stdio.h>

void permuter(int *tab, int a, int b)
{
  *tab[100] = *tab[a];
  *tab[a] = *tab[b];
  *tab[b] = *tab[100];
}

int inverse(int n)
{
  int tab[100], p = 1, test = n, c;

  while(test/10 != 0)
  {
    test /= 10;
    p++;
  }

  test = n;
  c = p/2;

  for (int i=p-1 ; i>=0 ; i--)
  {
      tab[i] = test % 10;
      test /= 10;
  }

  for(i=0 ; i<c ; i++)
  {
    permuter(tab, i, p-1-i);
  }

  int final = 0;
```

```c
  for(i=p-1 ; i>=0 ; i--)
  {
    final += tab[i]+10*i;
  }

  return final;
}

void palindrome(int n)
{
  if(n == inverse(n)) printf("%d est un palindrome.", n);
}

int main()
{
  printf("%d", inverse(1234));
  palindrome(12321);

  return 0;
}
```