

Exercice 1:

Qu'affiche le programme C suivant :

```
#include <stdio.h>
int main (void)
{
    int i;
    i=5;
    while (i){
        printf("La valeur de i est %d. \n", i);
        i--;}
    return(0);}
```

Exercice 2:

- Ecrire un programme C qui compte le nombre de caractères dans texte saisi (caractère par caractère) sur une seule ligne.
 - Par exemple, si le texte saisi est : "Bonjour je travaille."
 - le programme affichera : 21
- Reprendre l'exercice précédant en ne déclarant qu'une seule variable (de type unsigned char).

Exercice 3:

- Écrire un programme qui lit, caractère par caractère, une séquence de lettres différentes, puis les affiche dans l'ordre croissant (d'abord les minuscules, ensuite les majuscules). La lecture s'arrête lorsque l'utilisateur rentre le caractère '\$'.
 - Par exemple, si l'utilisateur entre "EaxcDA\$", le programme affichera "acxADE".
- Modifier le programme précédent de manière à générer une erreur si l'utilisateur rentre deux fois le même caractère.

Remarque : L'utilisation des tableaux, des pointeurs, des chaînes de caractères et de la récursivité n'est pas autorisé dans cet exercice. Il n'est pas permis d'utiliser plus de 5 variables pour résoudre cet exercice.

Exercice 4:

Ecrire un programme C qui :

- lit un nombre entier positif supérieur ou égal à 99. Si l'utilisateur rentre un nombre négatif, inférieur à 99 ou un caractère, le programme redemande une autre lecture à l'utilisateur jusqu'à ce que le nombre lu soit supérieur ou égal à 99;
- affiche les deux premiers chiffres du nombre lu. Par exemple, si le nombre lu est 325524, votre programme affichera 32.

Exercice 5:

Le but de cet exercice est d'abord d'illustrer que les algorithmes naïfs sont souvent suffisants pour traiter des problèmes simples ayant des petites entrées. En présence de problèmes complexes ou ayant des entrées de plus grandes tailles, il est nécessaire d'étudier les propriétés des problèmes considérés afin d'arriver à des algorithmes plus efficaces. Nous utiliserons dans cet exercice des problèmes simples (test de primalité et de vérification si un nombre est parfait ou non) pour illustrer notre observation. Il est conseillé d'utiliser le type "long unsigned int" pour représenter les entiers positifs de cet exercice.

1. Ecrire un programme C qui lit un entier positif n et affiche le résultat du calcul de l'expression $2^n - 1$. Les nombres de la forme $M_n = 2^n - 1$ sont appelés les nombres de Mersenne. Nous étudierons plus tard différents algorithmes pour le calcul des puissances de la forme a^b avec a et b des entiers.

Remarque 1 : Pensez aussi à utiliser les opérateurs bits à bits pour le calcul de M_n .

Remarque 2 : Quelles précautions à prendre pour éviter les débordements de valeurs ?

Remarque 3 : Y-a-il une différence entre les constantes entières suivantes "1", "1u", "1l", "1lu".

2. Ecrire un programme C qui lit un entier positif et vérifie s'il est premier ou non. Un nombre est dit premier s'il n'admet que deux diviseurs distincts : le nombre 1 (un) et le nombre lui-même. Par convention, le nombre 2 (deux) est considéré comme un nombre premier.
3. Ecrire un programme C qui lit un entier positif et vérifie s'il est parfait ou non. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs stricts (qui lui sont inférieurs). Voici la liste des premiers nombres parfaits :

- 1 $6=1+2+3$
- 2 $28=1+2+4+7+14$
- 3 496
- 4 8128
- 5 33550336
- 6 8589869056
- 7 137438691328
- 8 2305843008139952128
- 9 2658455991569831744654692615953842176

4. Ecrire un programme C qui lit un entier strictement positif p et affiche les p premiers nombres de Mersenne qui sont premiers. Un nombre est dit de Mersenne premier s'il est de la forme $2^n - 1$ et s'il n'accepte que deux diviseurs distincts. Tester votre algorithmes avec les différentes valeurs de p suivantes : $p=5$, $p=8$, $p=9$ et $p=15$.
5. Reprendre la question précédente (Question 4). A chaque fois qu'un nombre Mersenne ($M_n = 2^n - 1$) premier est trouvé, on affichera également l'expression $2^{n-1} \cdot (2^n - 1)$. Que remarquez-vous ?
6. Ecrire un programme C qui lit un entier strictement positif p et affiche le $p^{\text{ème}}$ élément de la suite suivante définie par :

$$\begin{cases} S(0) &= 4, \\ S(n+1) &= S(n)^2 - 2, \quad \text{si } n \geq 0. \end{cases}$$

7. Reprendre la question précédente (Question 4). A chaque fois qu'un nombre Mersenne ($M_n = 2^n - 1$) premier est trouvé, calculer puis afficher les valeurs de $S(n-2)$ et de l'expression $S(n-2) \% M_n$ (le reste de la division de $S(n-2)$ par M_n). Que remarquez-vous ?