Exercice 1.

A. *Soit la fonction récursive F d'un paramètre entier n suivante :*

```
entier F(n:entier)
  si n=0
     retourner 2
  sinon
     retourner F(n-1)*F(n-1)
```

i) *Que calcule cette fonction ? Le prouver.*

ii) *Déterminer la complexité de la fonction F. Comment améliorer cette complexité ?*

B. *Soit la fonction G suivante :*

```
entier G(n:entier)
R=cste
Pour i=1 à n faire
   R=R*R
Finpour
retourner R
```

i) *Que calcule cette fonction ? Le prouver.*

ii) *Déterminer la complexité de la fonction G.*

Exercice 2.
On considère un tableau A de n entiers. Soit k un entier donné qui n'appartient pas au tableau A.

A. *Ecrire en C les fonctions $minimum(A)$, $maximum(A)$, $successeur(A,k)$ et $predecesseur(A,k)$. La fonction $successeur(A,k)$ retournera l'indice de la case de A qui suit la première occurrence de k.*

B. *Reprendre la question précédente en supposant que le tableau A est trié en ordre croissant.*

Exercice 3.
On considère un tableau A de n éléments, que l'on suppose trié en ordre croissant. On cherche à construire un algorithme permettant de savoir à quel endroit du tableau se trouve une valeur cle. On supposera que cle est bien dans le tableau et on recherchera la première occurrence de cette valeur.

A. *Donner un algorithme itératif qui résout ce problème. Indiquer un invariant de boucle pour cet algorithme.*

B. *Donner les complexités O et Ω de cet algorithme.*

C. *Ecrire cet algorithme sous forme récursive et le prouver.*

D. *Comment modifier ces deux versions si l'on est pas sûr que cle appartienne au tableau ?*

Exercice 4.
On suppose que A est un talbeau trié de n éléments. On considère l'algorithme suivant :

```
entier Cherche_Dich_it(A: tableau, n:entier, clé: entier)
d=1
f=n
trouve=FAUX
Répéter
```

```
  i=(d+f)/2
  Si A[i]=clé
     trouve=VRAI
  Si A[i]<clé
     d=i+1
  Si A[i]>clé
     f=i-1
Tantque trouve=FAUX
retourner i
```

A. *Développer cet algorithme sur le tableau suivant avec clé=30*

$$T = [1, 7, 8, 9, 12, 15, 18, 22, 30, 31]$$

B. *Indiquer un invariant de boucle pour cet algorithme.*

C. *On suppose que le tableau* $[1 \dots n]$ *contient* $n = 2^k$ *éléments, où k est un entier positif. Combien d'itérations l'algorithme effectuera-t-il au maximum ? En déduire la complexité O de cet algorithme.*

D. *Pour* $k = 100$*, comparer les complexités des algorithmes de recherche séquentielle et dichotomique.*

E. *Ecrire l'algorithme sous forme récursive et l'éxécuter sur l'exemple.*

F. *Prouver la validité de la version récursive et donner sa complexité en O. Comment modifier ces versions de l'algorithme si l'on est pas sûr que cl appartienne au talbeau ?*