PYTHON. **Méthode de Karatsuba pour la multiplication des polynômes. Corrigé.**

1)

```
def produit(U,V) :
    n = len(U) ; W = [0]*n
    for k in tange(n) :
        for j in range(k+1) :
            W[k] += U[j]*V[k-j]
    return W
```

Remarque : On suppose que les deux listes `U` et `V` sont de même longueur n, et on suppose ici de plus que $\deg U + \deg V \leq n$. Le produit $W = UV$ est ainsi obtenu par : $\forall k \leq n$, $w_k = \sum_{j=0}^{k} u_j v_{k-j}$.
Dans le cas général, on a $\deg U + \deg V \leq 2n$, et on prendrait $\forall k \leq 2n$, $w_k = \sum_{j=\max(0,k-n)}^{\min(k,n)} u_j v_{k-j}$.

2) a)

```
def oppose(U) :
    n = len(U) ; V = [0]*n
    for i in range(n) :  V[i] = U[n-1-i]
    return V

def addition(U,V) :
    n = len(U) ; m = len(V)
    if n >= m :
        W = U.copy()     # On effectue une copie de U pour initialiser W.
        for i in range(m) :  W[i] = W[i] + V[i]
    return W
    else :  return addition(V,U)

def decalage(U,e) :  return [0]*e + U
```

Les fonctions `oppose` et `addition` demandent $O(n)$ opérations, où n est la longueur maximale.

La fonction `decalage` nécessite $O(n+e)$ opérations, où n est la longueur de la liste U.

b) On calcule $A = U_b V_b$, $B = U_h V_h$, et $C = (U_b + U_h)(V_b + V_h)$.

On a alors $UV = A + (C - A - B)X^e + BX^{2e}$.

c)

```
def partition U =
    n = len(U) ; e = n//2
```

```
    Ub = [ U[k] for k in range(e) ]
    Uh = [ U[k] for k in range(e,n) ]
    return Ub,Uh
```

Variante plus concise :

```
    return U[:e],U[e:]
```

d) Ne pas oublier le test d'arrêt dans la fonction récursive :

```
def produit U V =
    n = len(U)
    if  n == 0 :  return []
    if  n == 1 :  return [ U[0]*V[0] ]
    (Ub,Uh) = partition U ; (Vb,Vh) = partition V
    A = produit Ub Vb ; B = produit Uh Vh
    C = produit (somme Ub Uh) (somme Vb Vh)
    D = somme (somme C (oppose A)) (oppose B)
    E = somme (somme A (decalage D e)) (decalage B (2*e))
    return E
```

On procède par dichotomie (et récurrence).

En $O(n)$ opérations, on effectue les partitions de U et V. Et on calcule $U_b + U_h$ et $V_b + V_h$.

On effectue les produits A, B et C en $c(\lfloor \frac{n}{2} \rfloor) + 2c(\lceil \frac{n}{2} \rceil)$ opérations.

On en déduit UV en utilisant les fonctions `somme` et `decalage`, donc en $O(n)$ opérations.

3) a) $c\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$ correspond au coût de `(produit Ub Vb)`.
$c\left(\left\lceil \frac{n}{2} \right\rceil\right)$ correspond au coût de `(produit Uh Vh)` et du calcul de `C`.

Les autres opérations (additions, sommes, décalages) sont en $O(n)$.

Il existe ainsi une constante k telle que $c(n) \leq c\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 2c\left(\left\lceil \frac{n}{2} \right\rceil\right) + kn$, et on prend $c(1) = 1$.

b) Avec $n = 2^p$, on a $c(n) \leq 3c\left(\frac{n}{2}\right) + kn \leq 3^p c\left(\frac{n}{2^p}\right) + kn\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + ... + \left(\frac{3}{2}\right)^{p-1}\right)$.

Ainsi, $c(n) \leq (1+3k)\left(\frac{3}{2}\right)^p n = (1+3k)3^p$.

Comme $p = \log n$, on a $3^p = 3^{\log n} = \exp\left(\frac{\ln 3}{\ln 2}\ln n\right) = n^\alpha$, où $\alpha = \frac{\ln 3}{\ln 2}$.

On en conclut bien $c(n) \leq (1+3k)n^\alpha$.

c) On pose $p = E(\log n)$. On a $p \leq \log n < p+1$, donc $2^p \leq n < 2^{p+1}$.

Donc $c(n) \leq c(2^{p+1}) \leq (1+3k)(2^{p+1})^\alpha = (1+3k)2^\alpha(2^p)^\alpha \leq (1+3k)2^\alpha n^\alpha$.