

Exercice 5 :

1)

Le sous-tableau ayant la somme maximal est 5, -1, 1, 4

2)

ST_somme (t : tab, d : entier, g : entier) : entier

-> Entrées : t (le tableau auquel appartient le sous-tableau), d (l'indice de le plus grand du sous-tableau), g (l'indice le plus petit du sous-tableau)

-> Sortie : La fonction renvoie la somme des éléments du sous-tableau

-> Variables locales : s : entier;

Début

s <- 0;

Tant que g <= d faire

s <- s + t[g];

g <- g + 1;

Fin Tant que

Retourner s;

Fin

3)

Max_somme (t : tab) : entier

-> Entrée : t (le tableau dont on souhiate déterminer la plus grande somme)

-> Sortie : La fonction renvoie la plus grande somme trouvé

-> Variables locales : max, i, j : entier;

Début

s <- 0;

max <- t[0] + t[1];

Pour i de 0 à n-1 faire

Pour k de n à i+1 faire

Pour j de i à k faire

Si max < ST_somme (t, j, k)

max <- ST_somme (t, j, k);

Fin Si

j <- j + 1;

Fin Pour

k <- k - 1;

Fin Pour

i <- i + 1;

Fin Pour

Retourner max;

Fin

Exercice 6 :

1)

Somme (P1 : tab, P2 : tab) : tab

-> Entrées : P1 (un tableau contenant un 1er polynôme), P2 (un tableau contenant un 2ème polynôme)

-> Sortie : Retourne la variable locale P3 de type tab dans laquelle est stockée la somme de P1 et P2.

-> Variable locale : i : entier ; P3 : tab ;

Début

```

i <- 0 ;
Tant que i <= P1.Fin et i <= P2.Fin
    P3.T[i] = P1.T[i] + P2.T[i] ;
    i <- i + 1 ;
Fin Tant que
P3.Fin <- i ;
Retourner P3 ;

```

Fin

2)

Derivee (P1 : tab)

-> Entrée : P1 (un tableau contenant le polynôme que l'on veut dériver)

-> Sortie : P1 en sortie sera le dérivée de P1 en entrée

-> Variable locale : i : entier ;

Début

```

    Pour i de 0 à P1.Fin-1 faire
        P1.T[i] <- ( i + 1 ) * P1.T[i+1] ;
    i <- i + 1 ;
    Fin Pour
    P1.Fin <- P1.Fin - 1 ;

```

Fin

3)

Produit (P1 : tab, P2 : tab) : tab

-> Entrées : P1 (un tableau contenant un 1er polynôme), P2 (un tableau contenant un 2ème polynôme)

-> Sortie : Retourne la variable locale P3 de type tab dans laquelle est stockée le produit de P1 et P2.

-> Variable locale : i, j, k : entier ; P3 : tab ;

Début

```

    P3.Fin = P1.Fin + P2.Fin ;
    Pour i de 0 à P1.Fin faire
        Pour j de i à P2.Fin faire
            k <- i + j ;
            P3.T[k] <- P1.T[i] + P2.T[j] + P3.T[k] ;
            j <- j + 1 ;
        Fin Pour
        i <- i + 1 ;
    Fin Pour
    Retourner P3 ;

```

Fin

Exercice 7 :

1)

Iteration (n : entier) : entier

-> Entrée : n (un entier dont on veut savoir combien de chiffres il prend pour l'écrire en base 10)

-> Sortie : Renvoie la variable locale nb qui décrit la quantité de chiffres qu'il faut pour écrire n en base 10

-> Variables locales : nb, coef : entiers ;

Début

```

    coef <- 10 ;
    nb <- 1 ;

```

```

    Tant que n/coef != 0
        nb <- nb + 1 ;
        coef <- coef*10 ;
    Fin Tant que
    Retourner nb ;

```

Fin

Recuratif (n : entier, nb : entier, coef : entier) : entier

-> Entrées : n (un entier dont on veut savoir combien de chiffres il prend pour l'écrire en base 10)
 nb (le nombre de chiffre requis pour écrire n en base 10 / doit être égal à un quand la fonction est appelé pour la première fois), coef (sert pour les conditions en tant que puissance de 10 / doit être égal à 10 quand la fonction est appelé pour la première fois)

-> Sortie : Renvoie la variable locale nb qui décrit la quantité de chiffres qu'il faut pour écrire n en base 10

Début

```

    Si n/coef = 0
        Retourner nb ;
    Sinon
        Retourner Recursif ( n, nb+1, coef*10 );

```

Fin Si

}

2) Il y a nb affectations

Exercice 8 :

1)

Unimodal (t : tab) : booléen

-> Entrée : t (le tableau qu'on souhaite déterminer comme unimodal ou pas)

-> Sortie : La fonction Unimodal renvoie vrai si le tableau est unimodal ou faux s'il ne l'est pas.

-> Variables locales : i : entier;

Début

```

    i <- 0;
    Tant que t[i] <= t[i+1] faire
        i <- i + 1;
    Fin Tant que
    Pour i de i à n faire
        Si t[i] <= t[i+1]
            Retourner 0;
        Fin Si
        i <- i + 1;
    Fin Pour
    Retourner 1;

```

Fin

2)

MaxUni (t : tab) : entier

-> Entrée : t (le tableau unimodal dont on veut savoir le maximum)

-> Sortie : La fonction MaxUni renvoie la valeur de la variable locale i dans laquelle est stockée l'indice de la valeur du maximum

-> Variables locales : i : entiers;

Début

```

i <- 0;
Tant que t[i] <= t[i+1] faire
  i <- i + 1;
Fin Tant que
Retourner t[i];
}

```

3)

A la recherche du maximum :

- Dans un tableau trié, le maximum est $t[n-1]$. La complexité est donc $O(1)$.
- Dans un tableau unimodal, la complexité est $O(\log n)$
- Dans un tableau quelconque, la complexité est $O(n)$

A la recherche d'une valeur :

- Dans un tableau trié, on trouve le maximum avec une recherche dichotomique.

La complexité est donc de $O(\log n)$

- Dans un tableau unimodal, on a deux tableau triés : une première moitié trié en ordre croissant et une deuxième moitié trié en ordre décroissant. La complexité reste $O(\log n)$

- Dans un tableau quelconque, il faut comparer chaque élément. La complexité est $O(n)$.

Exercice 9 :

1)

TQ_doublons (t : tab)

-> Entrée : t (le tableau quelconque qu'on souhaite modifier en enlevant les doublons)

-> Sortie : Le tableau t est modifier à la fin de la fonction

-> Variables locales : i, j, k : entier;

Début

Pour i de 0 à n faire

Pour j de i+1 à n faire

Si $t[i] = t[j]$

$t[j] <- 0;$

$k <- j;$

Tant que $k \neq n-1$

$t[k] <- t[k+1];$

$k <- k + 1;$

Fin Tant que

$t.fin <- t.fin - 1;$

Fin Si

$j <- j + 1;$

Fin Pour

$i <- i + 1;$

Fin Pour

Fin

2)

TT_doublons (t : tab)

-> Entrée : t (le tableau trié qu'on souhaite modifier en enlevant les doublons)

-> Sortie : Le tableau t est modifier à la fin de la fonction

-> Variables locales : i, j : entier;

Début

```

Pour i de 0 à n-1 faire
  Si t[i] = t[i+1]
    t[i] <- 0;
    j <- i;
    Tant que j != n-1
      t[j] <- t[j+1];
      j <- j + 1;
    Fin Tant que
    t.fin <- t.fin - 1;
  Fin Si
  i <- i + 1;
Fin Pour
Fin

```

3) Oui. La complexité de TQ_doublons est d'au moins $O(n^2)$ tandis que la complexité de TT_doublons est $O(n \cdot \log n)$.