Type liste : ^ cel Type cel : eng Info : entier ; Svt : liste ; Fin	Type Arbre: ^ cel1 Type cel1 : eng Info : entier ; fg : arbre ; fd : arbre ; Fin	Type File : ^ cel2 Type cel1 : eng Info : Arbre; Svt : File ; Fin

X= 1234 1 2 3 4 4

```
Fct decomposee_nb(E/ x : entier, ES/ nbelt : entier ) : liste
Debut
   T, p : liste ;
   Reste : entier ;
   T    nil ;
   nbelt    0 ;
   TQ ( x ≠ 0)
     Faire
        Reste    x mod 10 ;
        Allouer(p) ;
         P^.info    reste ;
         P^.svt    t ;
         T    p ;
         nbelt    nbelt +1 ;
         X    x/ 10 ;
     Fait ;
    Retourner t ;
Fin
```

2/3/

```
Procedure tri_max(E/ t : liste)
Debut
 t1, adrmax : liste ;
 x : entier ;
   3  2  1
TQ( t^.svt  ≠ nil)
  Faire
    t1    t^.svt ;
    adrmax     t ;
    TQ( t1  ≠ nil)
      Faire
       Si(t1^.info > adrmax^.info)
                    alors
                       adrmax    t1 ;
       fsi ;
       t1    t1^.svt ;
    fait ;
    x    t^.info ;
    t^.info    adrmax^.info ;
    adrmax^.info    x;
```

```
Procedure tri_mim(E/ t : liste)
Debut
 t1, adrmin : liste ;
 x : entier ;
   3  2  1
TQ( t^.svt  ≠ nil)
  Faire
    t1    t^.svt ;
    adrmin     t ;
    TQ( t1  ≠ nil)
      Faire
       Si(t1^.info < adrmin^.info)
                    alors
                       adrmin    t1 ;
       fsi ;
       t1    t1^.svt ;
    fait ;
    x    t^.info ;
    t^.info    adrmin^.info ;
    adrmin^.info    x;
```

t t^.svt ; fait Fin	t t^.svt ; fait Fin

4/

```
Fonction calcul_nb_liste(t1 : liste, nb_elt : entier) :entier
Début
 Si (t1 ≠ nil) alors retourner (t1^.info * puis( 10, nb_elt-1)+ calcul_nb_liste(t^.svt, nb_elt-1)) ;
                Retourner 0 ;
Fin
```

5/

```
Fonction recherche_val(E/ r : arbre,  E/ x : entier) :  entier
Début
  Si (r ≠ nil) alors
                 Si (r^.info = x) alors retourner 1 ;
                                  sinon retourner (recherche_val(r^.fg, x) +  recherche_val(r^.fd, x) ) ;
               Sinon retourner 0 ;
Fin
```

6/

```
procedure affiche_arbre(E/ r : arbre)
Début
 TQ(r^.fg ≠ nil  et r^.fd ≠ nil)
   Faire
     Ecrire(r^.info) ;
     Ecrire(r^.fg^.info) ;
     Ecrire(r^.fd^.info) ;
     r  r^.fg^.fd ;
  Fait
  Ecrire(r^.info) ;
Fin
```

7/

```
procedure affiche_cycle(E/ r : arbre)
Début
   TQ(r^.fg ≠ nil  et r^.fd ≠ nil)
   Faire
         Ecrire(r^.info) ;
         r   r^.fg^.fd ;
      fsi ;
   fait
   Ecrire(r^.info) ;
Fin
```

8/

Procedure supprime_liste(ES/ t : liste) Début tmp : liste ; TQ(t ≠ nil) Faire tmp t ; t t^.svt ; Liberer (tmp) ; Fait ; Fin	FONCTION inser_elt(E/ x :entier) : Arbre Début P : arbre ; Allouer (p) ; P^.info x ; P ^.fg nil ; P ^.fd nil ; Retourn p ; Fin

9/

Début

```
Lire (N) ;
Racine   inser_elt(N) ;     lg   1 ;

Enfiler ( t, q, racine) ;
Trouv   0;
TQ(t^. info^.info  ≠  0  et  trouv = 0)
   Faire
     Defiler(t,q, tmp) ;

     lmin   decompose_nb(tmp ^.info, nb_elt_min) ;
     tri_min(lmin) ;
     min    calcul_nb_liste(lmin, nb_elt_min)) ;
     tmp^.fg    inser_elt(min) ;

     lmax   decompose_nb(tmp ^.info, nb_elt_max) ;
     tri_max(lmax) ;
     max    calcul_nb_liste(lmax, nb_elt_max)) ;
     tmp^.fd    inser_elt(max) ;

     tmpg   temp^.fg ;
    tmpd   temp^.fd ;

    D   max – min ;

     ldiff   decompose_nb(D, nbDiff) ;
     tri_max(ldiff) ;
     maxDif    calcul_nb_liste(ldiff, nbDiff)) ;
```

```
    Trouv     recherche_val(racine,   maxDif) ;
    tmpg^. fd      inser_elt(D) ;
    tmpd^.fg    Tmpg^. fd ;
     lg    lg+1 ;
     lmin    supprime_liste(lmin) ;
     lmax    supprime_liste(lmax) ;
     Enfiler ( t, q, tmpg^.fd) ;
  fait
   Defiler(t,q, tmp) ;
    affiche_arbre(racine) ;
    Ecrire(lg) ;
    affiche_cycle(racine) ;

Fin
```