// FOURNI

```
typedef struct noeud {          // Type noeud
     int info;
     struct noeud* sag;
     struct noeud* sad;
} noeud;

typedef noeud* arbre;           // Type arbre


typedef struct maillon {        // Type maillon
     int info;
     struct maillon* succ;
} maillon;

typedef maillon* liste;         // Type liste

typedef liste pile;             // Type pile

typedef struct file {           // Type file
     maillon* entree;
     maillon* sortie;
     unsigned int taille;
} file;
```

// FOURNI (arbres binaires)

```
arbre arbret(unsigned int n, const int t[]);  // convertit un tableau en arbre
string chainea(arbre a); // Représente un arbre sous forme de chaîne de caractères
arbre singleton(int v);  // Crée un arbre d'un seul nœud de valeur v donnée
```

// FOURNI (listes, piles, files)

```
// ...
```

// A FAIRE (arbres binaires qcq)

```
int nbNoeuds(arbre a) ;
int hauteur(arbre a) ;
int minimum(arbre a) ;
arbre copie(arbre a) ;
void miroir(arbre a) ;
int estParfait(arbre a) ;
int profondeurSommeMax(arbre a) ;
liste listeFeuilles(arbre a) ;
int detruire(arbre* a) ;
```

// A FAIRE (arbres binaires de recherche : G <= R < D)

```
int estABR(arbre a, arbre inf = NULL, arbre sup = NULL) ;
noeud* chercher(arbre a, int v) ;
int inserer(arbre* a, int v) ;
int supprimer(arbre* a, int v) ;
```

// A FAIRE

```
int main(int argc, char* argv[]) ;    // des schémas de tests sont fournis
```