Es.1.

Si definisca una funzione ricorsiva C che restituisce 0 se il numero di nodiinterni è uguale al numero delle foglie e un valore diverso da zero altrimenti. La funzione ha il seguente prototipo

int FoUgInt(TreePtr tPtr)

Es.2.

Si definisca una funzione C che restituisce il numero dei nodi radici di sottoalberi in cui il numero di nodi interni è uguale al numero delle foglie. La funzione ha il seguente prototipo

int nFoUgInt(TreePtr tPtr)

Es 3. Si definisca una funzione C che verifica se la frontiera dell'albero in input è ordinata. La frontiera di un albero è la sequenza delle foglie da sinistra verso destra, qui intendiamo la sequenza di interi contenuti nelle foglie, da sinistra verso destra.
La funzione ha il seguente prototipo e restituisce 1 se la frontiera è ordinata.

int **frontOrd**(TreePtr t);

Soluzioni.

```
int FoUgInt(TreePtr tPtr)
/*restituisce la differenza tra  il numero di nodi interni
quello delle foglie*/
{
if (!tPtr) return 0;
if (!tPtr->lPtr && !tPtr->rPtr) return -1;
return FoUgInt(tPtr->lPtr)+FoUgInt(tPtr->rPtr) +1;
}


int nFoEqInt(TreePtr tPtr)
/*restituisce il numero di nodi radici di sottoalberi
con un numero di foglie uguali al numero dei nodi interni*/
{int ris =0;
nFoUIntAus(tPtr,&ris);
return ris;}

int nFoUIntAus(TreePtr tPtr, int * ris)
/*restituisce la differenza tra il numero dei nodi interni e il
 numero delle  foglie in t, in ris mette il numero di nodi radici di sottoalberi
con un numero di foglie uguali al numero dei nodi interni*/
{int diff;
if (!tPtr) return 0;
if (foglia(tPtr)) return -1;
diff = nFoUIntAus(tPtr->lPtr,ris) + nFoUIntAus(tPtr->rPtr,ris) +1;
if (diff == 0) (*ris)++;
return diff;}

int frontOrd(TreePtr  t)
/*postc: restituisce 1 se la frontiera di interi, letta da sinistra a destra e'
ordinata in modo crescente*/
{int n = INT_MIN; /* è il minimo intero rappresentabile, definito in
<limits.h> */
if (!t) return 1;
else return frontOrdAus(t,&n);}

int frontOrdAus(TreePtr  t, int* k)
/*k per il valore dell'ultima foglia visitata
```

```
*prec: t != NULL
*postc: restituisce 1 se la frontiera di interi, letta da sinistra a destra e'
ordinata in modo crescente*/
{int ris = 1;
if (foglia(t))
        if (t->elem >= *k) {*k = t->elem; return 1;} else return 0;
if (t->left) ris = frontOrdAus(t->left,k);
if (ris && t->right) ris = frontOrdAus(t->right,k);
return ris;}
```

Qui si evita di visitare tutto l'albero se la frontiera non è ordinata.
In alternativa, visitando tutto l'albero:

```
int frontOrd(TreePtr  t)
/*postc: restituisce 1 se la frontiera di interi, letta da sinistra a destra e'
ordinata in modo crescente*/
{int n = INT_MIN; /* è il minimo intero rappresentabile, definito in
<limits.h> */
return frontOrdAus(t,&n);}

int frontOrdAus(TreePtr  t, int* k)
/*k per il valore dell'ultima foglia visitata
*postc: restituisce 1 se la frontiera di interi, letta da sinistra a destra e'
ordinata in modo crescente*/
{if (!t) return  0;
if (foglia(t))
        if (t->elem >= *k) {*k = t->elem; return 1;} else return 0;
return frontOrdAus(t->left,k) &&frontOrdAus(t->right,k);
}
```