```
/*Esercizio 2.
Si definisca una funzione C che, dato  in input un albero binario di interi,
restituisce 1 se l'albero è dominante.
Un albero binario si dice dominante se ogni nodo dell'albero domina
il sottoalbero in esso radicato, 0 altrimenti.
Un nodo domina il proprio sottoalbero se l'intero
in esso contenuto è maggiore della somma  degli interi  contenuti nei nodi del sottoalbero
(o equivalentemente la somma dei contenuti dei nodi nel sottoalbero sinistro e destro).
La funzione ha il seguente prototipo:
int domSA(TreePtr t)
Non si dimentichi di precisare pre e post condizioni!
*/
 int sottDom(TreePtr t)
/*postc: restituisce 1 se per ogni  sottoalbero radicato in n di t,
la somma dei valori interi nei nodi e' minore del valore di n, 0 altrimenti
*/
{int n,ris=1;
if (!t) return 1;
else {n = sottDomAus(t,&ris);
return ris;
/* escludiamo il caso dell'albero vuoto perchè se restituissimo 0
nella sottDomAus sull'albero vuoto, su un albero con un solo nodo
di contenuto 0, la funzione retstituirebbe 0 e non 1*/
}}


 int sottDomAus(TreePtr t,int* ris)
/*prec: t != NULL;
postc: restituisce la somma dei valori nei nodi e pone
ris a 0 non appena incontra un nodo che non domina il  albero in cui è radicato,
alrimenti ris resta invariato*/
{ int risPar=0;
if (!(t -> left) && !(t -> right)) return t->elem;
if (*ris && t -> left)  risPar = sottDomAus(t -> left,ris);
if ( *ris && t-> right) risPar += sottDomAus(t -> right,ris);
if (*ris && risPar < t -> elem) return risPar + t -> elem;
else {*ris =0; return 0;}
}
```