```c
/*
 * Much of this code is adapted from "Algorithms in C, Third Edition,"
 * by Robert Sedgewick, Addison Wesley Longman, 1998, and from
 * Sedgewick’s own Java code for left-leaning red-black trees.
 */
#include <string.h>
#include <stdlib.h>
#include "Item.h"

/*
  Constantes
*/
const char no_externo[] = "\\Tn";
const char no_interno[] = "\\Tc*{1.5pt}";
const char tree_sep[] = ".3mm";
const char level_sep[] = "2.5mm";

/*
 * Back to Sedgewick stuff
 */

typedef struct STnode* link;
struct STnode { Item item; link l, r; int N, red; };

static link head, z;

link NEW(Item item, link l, link r, int N, int red)
  { link x = malloc(sizeof *x);
    x->item = item; x->l = l; x->r = r; x->N = N; x->red = red;
    return x;
  }

void STinit()
  { head = (z = NEW(NULLitem, 0, 0, 0, 0)); }

int STcount() { return head->N; }

#define hl  (h->l)
#define hr  (h->r)
#define hll  (h->l->l)

Item searchR(link h, Key v)
  { Key t = key(h->item);
    if (h == z) return NULLitem;
    if eq(v, t) return h->item;
    if less(v, t) return searchR(hl, v);
             else return searchR(hr, v);
  }
Item STsearch(Key v)
  { return searchR(head, v); }

link fixNr(link h)
  { h->N = hl->N + hr->N +1;
    return h;
  }

link rotL(link h)
  { link x = hr; hr = x->l; x->l = fixNr(h);
    x->red = x->l->red; x->l->red = 1;
    return fixNr(x);
  }

link rotR(link h)
  { link x = hl; hl = x->r; x->r = fixNr(h);
```

```
    x->red = x->r->red; x->r->red = 1;
    return fixNr(x);
  }

void colorFlip(link h)
  { h->red = 1 - h->red;
    hl->red = 1 - hl->red;
    hr->red = 1 - hr->red;
  }

link LLRBinsert(link h, Item item)
  { Key v = key(item);
    /* Insert a new node at the bottom*/
    if (h == z) return NEW(item, z, z, 1, 1);

    if (less(v, key(h->item)))
      hl = LLRBinsert(hl, item);
    else
      hr = LLRBinsert(hr, item);

    /* Enforce left-leaning condition */
    if (hr->red && !hl->red) h = rotL(h);
    if (hl->red && hll->red) h = rotR(h);
    if (hl->red && hr->red) colorFlip(h);

    return fixNr(h);
  }

void STinsert(Item item)
  { head = LLRBinsert(head, item); head->red = 0; }

void sortR(link h, void(*visit)(Item))
{
  if (h!=z) {
    sortR(hl, visit);
    visit(h->item);
    sortR(hr, visit);
  }
}

void STsort(void(*visit)(Item))
{ sortR(head, visit); }

/*
 * Integrity check
 */






/* Drawing? */

/* faz_desenho(): deveria chamar faca_desenho();
   acho que nao deveria devolver valor nenhum...
*/

int faz_desenho(link h, int nivel)
{
  int d1, d2;

  if (h->N == 0) {
    printf("%s\n", no_externo);
    return 1;
  }

  if (nivel == 0)
    printf("\\pstree[levelsep=%s, treesep=%s]{%s}{\n", level_sep,
           tree_sep, no_interno);
  else
    printf("\\pstree{%s}{\n", no_interno);
  d1 = faz_desenho(h->l, nivel+1);
  d2 = faz_desenho(h->r, nivel+1);
  printf("}\n");
  return d1 + d2 + 1;
}

void STdraw()
{
  printf("\\rput{90}{");
  faz_desenho(head, 0);
  printf("}\n");
}
```