```
/*
 * This code is from "Algorithms in C, Third Edition,"
 * by Robert Sedgewick, Addison Wesley Longman, 1998.
 * Com adicoes e edicoes de yk
 */
#include <stdlib.h>
#include "Item.h"

typedef struct STnode* link;
struct STnode { Item item; link l, r; int N; };
static link head, z;

/*
  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
*/

link NEW(Item item, link l, link r, int N)
  { link x = malloc(sizeof *x);
    x->item = item; x->l = l; x->r = r; x->N = N;
    return x;
  }
void STinit()
  { head = (z = NEW(NULLitem, 0, 0, 0)); }
int STcount() { return head->N; }
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(h->l, v);
             else return searchR(h->r, v);
  }
Item STsearch(Key v)
  { return searchR(head, v); }

/* prog12.11.c */

link rotR(link h)
  { link x = h->l; h->l = x->r; x->r = h;
    h->N = h->l->N + h->r->N + 1;
    x->N = x->l->N + x->r->N + 1;
    return x; }
link rotL(link h)
  { link x = h->r; h->r = x->l; x->l = h;
    h->N = h->l->N + h->r->N + 1;
    x->N = x->l->N + x->r->N + 1;
    return x; }

/* prog12.14.c */

link partR(link h, int k)
  { int t = h->l->N;
    if (t > k )
      { h->l = partR(h->l, k); h = rotR(h); }
    if (t < k )
      { h->r = partR(h->r, k-t-1); h = rotL(h); }
    return h;
```

```
  }

/* prog12.12.c */

link insertT(link h, Item item)
  { Key v = key(item);
    if (h == z) return NEW(item, z, z, 1);
    if (less(v, key(h->item)))
      { h->l = insertT(h->l, item); h = rotR(h); }
    else
      { h->r = insertT(h->r, item); h = rotL(h); }
    return h;
  }

/* prog13.2.c */

link insertR(link h, Item item)
  { Key v = key(item), t = key(h->item);
    if (h == z) return NEW(item, z, z, 1);
    if (rand() < RAND_MAX/(h->N + 1))
      return insertT(h, item);
    if less(v, t) h->l = insertR(h->l, item);
              else h->r = insertR(h->r, item);
    (h->N)++; return h;
  }
void STinsert(Item item)
  { head = insertR(head, item); }

/* Program 12.8 */

void sortR(link h, void (*visit)(Item))
  {
    if (h == z) return;
    sortR(h->l, visit);
    visit(h->item);
    sortR(h->r, visit);
  }
void STsort(void (*visit)(Item))
  { sortR(head, visit); }

/* /\* prog13.3.c *\/ */

/* link STjoinR(link a, link b) */
/*   {  */
/*     if (a == z) return b; */
/*     b = STinsert(b, a->item);  */
/*     b->l = STjoin(a->l, b->l);  */
/*     b->r = STjoin(a->r, b->r);  */
/*     fixN(b); free(a); */
/*     return b; */
/*   } */
/* link STjoin(link a, link b) */
/*   {  */
/*     if (rand()/(RAND_MAX/(a->N+b->N)+1) < a->N) */
/*          STjoinR(a, b); */
/*     else STjoinR(b, a); */
/*   } */

/* prog13.4.c */

link joinLR(link a, link b)
  {
    if (a == z) return b;
    if (b == z) return a;
```

```
    if (rand()/(RAND_MAX/(a->N+b->N)+1) < a->N)
         { a->r = joinLR(a->r, b);
           a->N = a->l->N + a->r->N + 1;
           return a; }
    else { b->l = joinLR(a, b->l);
      b->N = b->l->N + b->r->N + 1;
      return b; }
  }

link deleteR(link h, Key v)
  { link x; Key t = key(h->item);
    if (h == z) return z;
    if (less(v, t)) h->l = deleteR(h->l, v);
    if (less(t, v)) h->r = deleteR(h->r, v);
    if (eq(v, t))
      { x = h; h = joinLR(h->l, h->r); free(x); }
    if (h != z)
      h->N = h->l->N + h->r->N + 1;
    return h;
  }
void STdelete(Key v)
  { head = deleteR(head, v); }

/* prog12.13.c */

Item selectR(link h, int k)
  { int t = h->l->N;
    if (t > k) return selectR(h->l, k);
    if (t < k) return selectR(h->r, k-t-1);
    return h->item;
  }
Item STselect(int k)
  { return selectR(head, k); }

/* Deletion of rank r element */

void STdelete_r(int r)
{
  Item item = STselect(r);
  STdelete(key(item));
}

/* prog13.1.c */

link balanceR(link h)
  {
    if (h->N < 2) return h;
    h = partR(h, h->N/2);
    h->l = balanceR(h->l);
    h->r = balanceR(h->r);
    return h;
  }

void STbalance()
  { head = balanceR(head); }

/* 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");
}
```