```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#include "mapa.h"
#include "arvore.h"

#define TRUE 1

struct smapa {
  int chave;
  int dado;
  char vermelho;
  struct smapa* esq;
  struct smapa *dir;
};

typedef enum result {OK, RED, LEFTRED, RIGHTRED} Result;
static char *strstatus[] = {"ok", "red", "leftred", "rightred"};

static Mapa*cria_no (int c, int novodado);
static Mapa* rotaciona_dir(Mapa *m);
static Mapa* rotaciona_esq(Mapa *m);
static void trocaCores (Mapa *m);
static Mapa* corrigeDir (Mapa *m, Result* status);
static Mapa* corrigeEsq (Mapa *m, Result* status);
static Mapa* insereRec (Mapa* m, int chave, int novodado, Result* status);

static Mapa *cria_no (int c, int novodado) {
  Mapa *m = (Mapa *)malloc(sizeof(Mapa));
  if (m!=NULL) {
    m->esq = m->dir = NULL;
    m->chave =c;
    m->vermelho = TRUE;
    m->dado = novodado;
  }
  return m;
}

Mapa* cria (void) {
  return NULL;
}

void destroi (Mapa *m) {
  if (m!= NULL) {
    destroi (m->esq);
    destroi (m->dir);
    free(m);
  }
}

int busca (Mapa *m, int chave) {
  if (m == NULL) {
    return -1;
  }
  else if (m->chave == chave) {
    return m->dado;
  }
  else if (chave < m->chave) {
  return busca(m->esq, chave);
  }
  else if (chave > m->chave) {
    return busca(m->dir, chave);
  }
  else {
    return -1;
  }
}


Mapa* insere (Mapa* m, int chave, int novodado) {
  Result status;
  m = insereRec (m, chave, novodado, &status);
  if (status == RED) m->vermelho = 0;
  else if (status != OK) {
    printf ("erro ao voltar para a raiz: status invalido %s !\n",
             strstatus[status]);
    mostra(m);
    exit(1);
  }
  return m;
}

void mostra (Mapa* m) {
```

```
85   printf("[");
86   if (m != NULL) {
87     printf("<%d - %c> ", m->chave,(m->vermelho) ?  'r' : 'b');
88     mostra(m->esq);
89     mostra(m->dir);
90   }
91   printf("] ");
92 }
93
94 static Mapa* rotaciona_dir(Mapa *m) {
95   Mapa* esq = m->esq;
96   m->esq = esq->dir;
97   esq->dir = m;
98   return esq;
99 }
100
101 static Mapa* rotaciona_esq(Mapa *m) {
102   Mapa* dir = m->dir;
103   m->dir = dir->esq;
104   dir->esq = m;
105   return dir;
106 }
107
108 static void trocaCores (Mapa *m) {
109   char corpai = m->vermelho;
110   m->vermelho = !corpai;     /* troca a cor do pai */
111   m->esq->vermelho = corpai; /* os filhos recebem a cor inversa da do pai */
112   m->dir->vermelho = corpai;
113 }
114
115 static Mapa* corrigeEsq (Mapa *m, Result* status) {
116   switch (*status) {
117     case OK:  /* nada a corrigir */
118       break;
119     case RED: /* filho vermelho */
120       if (m->vermelho) *status = LEFTRED; /* nÃ³ vermelho, filho vermelho Ã  esquerda */
121       else *status = OK; /* nÃ³ preto, filho vermelho estÃ¡ ok */
122       break;
123     case LEFTRED: /* LL */
124       if (!m->dir || !m->dir->vermelho) {  /* filho direito preto: LLb */
125         m = rotaciona_dir(m);
126         trocaCores(m);
127         *status = OK;
128       }
129       else { /* filho direito vermelho: LLr */
130         trocaCores(m);
131         *status = RED;
132
133       }
134       break;
135     case RIGHTRED: /* LR */
136       if (!m->dir || !m->dir->vermelho) {  /* filho direito preto: LRb */
137         m->esq = rotaciona_esq(m->esq);
138         m = rotaciona_dir(m);
139         trocaCores(m);
140         *status = OK;
141       }
142       else { /* filho direito vermelho -> LRr */
143         trocaCores(m);
144         *status = RED;
145       }
146       break;
147   }
148   return m;
149 }
150
151 static Mapa* corrigeDir (Mapa *m, Result* status) {
152   switch (*status) {
153     case OK: /* nada a corrigir */
154       break;
155     case RED: /* filho vermelho */
156       if (m->vermelho) *status = RIGHTRED;
157       else *status = OK;
158       break;
159     case RIGHTRED: /* RR */
160       if (!m->esq || !m->esq->vermelho) { /* filho esquerdo preto: RRb */
161         m = rotaciona_esq(m);
162         trocaCores(m);
163         *status = OK;
164         /* completar */
165       }
166       else { /* filho esquerdo vermelho: RRr */
167         /* completar */
168         trocaCores(m);
169         *status = RED;
```

```
      }
      break;
    case LEFTRED: /* RL */
      if (!m->esq || !m->esq->vermelho) { /* filho esquerdo preto: RLb */
        m->dir = rotaciona_dir(m->dir);
        m = rotaciona_esq(m);
        trocaCores(m);
        *status = OK;
        /* completar */
      }
      else { /* filho esquerdo vermelho: RLr */
        trocaCores(m);
        *status = RED;
        /* completar */
      }
      break;
  }
  return m;
}

static Mapa* insereRec (Mapa* m, int chave, int novodado, Result* status){
  if (m==NULL) {
    m = cria_no (chave, novodado);
    *status = RED;
  }
  else if (chave < m->chave) {
    m->esq = insereRec (m->esq, chave, novodado, status);
    m = corrigeEsq (m, status);
  }
  else if (chave > m->chave) {
    m->dir = insereRec (m->dir, chave, novodado, status);
    m = corrigeDir (m, status);
  }
  return m;
}

```