#include <stdio.h>

#include <stdlib.h>

struct no

{

int info;

struct no \*esq, \*dir;

};

typedef struct no No;

No \*remove\_no(No \*raiz, int info);

No \*antecessor(No \*q);

No \*sucessor(No \*q, No \*\*pai);

No \*busca\_no(No \*raiz, int info, No \*\*pai);

void em\_ordem(No \*raiz);

void pre\_ordem(No \*raiz);

No \*insere\_no\_abb(No \*raiz, int info);

No \*insere\_no\_abb\_rec(No \*raiz, No \*novo);

No \*cria\_no(int info);

int main()

{

No\* raiz=NULL;

raiz = insere\_no\_abb(raiz, 30);

raiz = insere\_no\_abb(raiz, 10);

raiz = insere\_no\_abb(raiz, 20);

raiz = insere\_no\_abb(raiz, 5);

raiz = insere\_no\_abb(raiz, 8);

raiz = insere\_no\_abb(raiz, 25);

raiz = insere\_no\_abb(raiz, 40);

raiz = insere\_no\_abb\_rec(raiz, cria\_no(50));

em\_ordem(raiz);

//raiz = remove\_no(raiz,8);

raiz = remove\_no(raiz,10);

printf("\n");

em\_ordem(raiz);

return 0;

}

No \*cria\_no(int info)

{

No \*novo = malloc(sizeof(No));

novo->info = info;

novo->esq = novo->dir = NULL;

return novo;

}

No \*insere\_no\_abb\_rec(No \*raiz, No \*novo)

{

if (raiz == NULL)

raiz = novo;

else

if (novo->info < raiz->info)

raiz->esq = insere\_no\_abb\_rec(raiz->esq, novo);

else

raiz->dir = insere\_no\_abb\_rec(raiz->dir, novo);

return raiz;

}

No \*insere\_no\_abb(No \*raiz, int info)

{

No \*novo = cria\_no(info);

No \*p = NULL, \*q = NULL;

if (raiz == NULL)

{

raiz = novo;

}

else

{

q = raiz;

while (q != NULL)

{

p = q;

if (info < q->info)

q = q->esq;

else

q = q->dir;

}

if (info < p->info)

p->esq = novo;

else

p->dir = novo;

}

return raiz;

}

void pre\_ordem(No \*raiz)

{

if (raiz != NULL)

{

printf("%d ", raiz->info);

pre\_ordem(raiz->esq);

pre\_ordem(raiz->dir);

}

}

void em\_ordem(No \*raiz)

{

if (raiz != NULL)

{

em\_ordem(raiz->esq);

printf("%d ", raiz->info);

em\_ordem(raiz->dir);

}

}

No \*busca\_no(No \*raiz, int info, No \*\*pai)

{

No \*q = raiz;

\*pai = NULL;

while (q != NULL && q->info != info)

{

\*pai = q;

if (info < q->info)

q = q->esq;

else

q = q->dir;

}

return q;

}

No \*sucessor(No \*q, No \*\*pai)

{

\*pai = q;

q = q->dir;

while (q->esq != NULL)

{

\*pai = q;

q = q->esq;

}

return q;

}

No \*antecessor(No \*q)

{

q = q->esq;

while (q->dir != NULL)

{

q = q->dir;

}

return q;

}

No \*remove\_no(No \*raiz, int info)

{

No \*pai, \*filho;

No \*q = busca\_no(raiz, info, &pai);

No \*suc;

int flag\_rem = 1;

while (flag\_rem)

{

// tem duas subarvores

if (q->dir != NULL && q->esq != NULL)

{

suc = sucessor(q, &pai);

//printf("suc %d ", suc->info);

q->info = suc->info;

q = suc;

}

else

{

// sem filhos

if (q->dir == NULL && q->esq == NULL)

{

if (pai != NULL)

{

if (pai->dir == q)

pai->dir = NULL;

else

pai->esq = NULL;

}

else

raiz = NULL;

free(q);

flag\_rem = 0;

}

else

{

if (q->esq != NULL)

filho = q->esq;

else

filho = q->dir;

if (pai != NULL)

{

if (pai->dir == q)

pai->dir = filho;

else

pai->esq = filho;

}

else

raiz = filho;

free(q);

flag\_rem = 0;

}

}

}

return raiz;

}