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

#define MAX 4
#define MIN (MAX/2)

typedef struct {
    int id;
    char nome[50];
    int idade;
} Registro;

typedef struct NoArvoreB {
    int num_chaves;
    int chaves[MAX];
    long posicoes[MAX];
    struct NoArvoreB *filhos[MAX + 1];
    int folha;
} NoArvoreB;

FILE *arquivoDados;
int id_atual = 1000;

NoArvoreB* criarNo(int folha) {
    NoArvoreB* novoNo = (NoArvoreB*) malloc(sizeof(NoArvoreB));
    if (novoNo == NULL) {
        printf("Erro ao alocar memória para o nó da árvore.\n");
        exit(1);
    }
    novoNo->num_chaves = 0;
    novoNo->folha = folha;
    for (int i = 0; i <= MAX; i++) {
        novoNo->filhos[i] = NULL;
    }
    return novoNo;
}

long buscar(NoArvoreB* no, int id) {
    int i = 0;
    while (i < no->num_chaves && id > no->chaves[i]) {
        i++;
    }
```

```
    if (i < no->num_chaves && id == no->chaves[i]) {
        return no->posicoes[i];
    }

    if (no->folha) {
        return -1;
    }

    return buscar(no->filhos[i], id);
}

void dividirFilho(NoArvoreB* pai, int i, NoArvoreB* filho) {
    NoArvoreB* novoFilho = criarNo(filho->folha);
    novoFilho->num_chaves = MIN;

    for (int j = 0; j < MIN; j++) {
        novoFilho->chaves[j] = filho->chaves[j + MIN + 1];
        novoFilho->posicoes[j] = filho->posicoes[j + MIN + 1];
    }

    if (!filho->folha) {
        for (int j = 0; j <= MIN; j++) {
            novoFilho->filhos[j] = filho->filhos[j + MIN + 1];
        }
    }

    filho->num_chaves = MIN;

    for (int j = pai->num_chaves; j >= i + 1; j--) {
        pai->filhos[j + 1] = pai->filhos[j];
    }

    pai->filhos[i + 1] = novoFilho;

    for (int j = pai->num_chaves - 1; j >= i; j--) {
        pai->chaves[j + 1] = pai->chaves[j];
        pai->posicoes[j + 1] = pai->posicoes[j];
    }

    pai->chaves[i] = filho->chaves[MIN];
    pai->posicoes[i] = filho->posicoes[MIN];
    pai->num_chaves++;
}
```

```
void inserirNaoCheio(NoArvoreB* no, int id, long pos) {
    int i = no->num_chaves - 1;

    if (no->folha) {
        while (i >= 0 && id < no->chaves[i]) {
            no->chaves[i + 1] = no->chaves[i];
            no->posicoes[i + 1] = no->posicoes[i];
            i--;
        }
        no->chaves[i + 1] = id;
        no->posicoes[i + 1] = pos;
        no->num_chaves++;
    } else {
        while (i >= 0 && id < no->chaves[i]) {
            i--;
        }

        if (no->filhos[i + 1]->num_chaves == MAX) {
            dividirFilho(no, i + 1, no->filhos[i + 1]);

            if (id > no->chaves[i + 1]) {
                i++;
            }
        }
        inserirNaoCheio(no->filhos[i + 1], id, pos);
    }
}

void inserir(NoArvoreB** raiz, int id, long pos) {
    NoArvoreB* r = *raiz;
    if (r->num_chaves == MAX) {
        NoArvoreB* s = criarNo(0);
        *raiz = s;
        s->filhos[0] = r;
        dividirFilho(s, 0, r);
        inserirNaoCheio(s, id, pos);
    } else {
        inserirNaoCheio(r, id, pos);
    }
}

long salvarRegistro(Registro reg) {
    if (arquivoDados == NULL) {
        printf("Erro: arquivo não está aberto!\n");
```

```
        exit(1);
    }
    fseek(arquivoDados, 0, SEEK_END);
    long pos = ftell(arquivoDados);
    fwrite(&reg, sizeof(Registro), 1, arquivoDados);
    fflush(arquivoDados);
    return pos;
}

Registro lerRegistro(long pos) {
    Registro reg;
    if (arquivoDados == NULL) {
        printf("Erro: arquivo não está aberto!\n");
        exit(1);
    }
    fseek(arquivoDados, pos, SEEK_SET);
    fread(&reg, sizeof(Registro), 1, arquivoDados);
    return reg;
}

void fecharArquivo() {
    if (arquivoDados != NULL) {
        fclose(arquivoDados);
        arquivoDados = NULL;
    }
}

Registro coletarDadosUsuario() {
    Registro reg;
    reg.id = id_atual++;
    printf("ID gerado automaticamente: %d\n", reg.id);
    printf("Insira o nome: ");
    scanf(" %[^\n]%*c", reg.nome);
    printf("Insira a idade: ");
    scanf("%d", &reg.idade);
    return reg;
}

int main() {
    arquivoDados = fopen("registros.bin", "wb+");
    if (arquivoDados == NULL) {
        printf("Erro ao abrir o arquivo!\n");
        return 1;
    }
```

```
    NoArvoreB* raiz = criarNo(1);

    for (int i = 0; i < 3; i++) {
        printf("\nInserindo registro %d\n", i + 1);
        Registro novoRegistro = coletarDadosUsuario();
        long pos = salvarRegistro(novoRegistro);
        inserir(&raiz, novoRegistro.id, pos);
    }

    int id_busca;
    printf("\nDigite o ID para buscar: ");
    scanf("%d", &id_busca);

    long pos = buscar(raiz, id_busca);
    if (pos != -1) {
        Registro encontrado = lerRegistro(pos);
        printf("Registro encontrado: ID = %d, Nome = %s, Idade = %d\n",
encontrado.id, encontrado.nome, encontrado.idade);
    } else {
        printf("Registro com ID %d não encontrado.\n", id_busca);
    }

    fecharArquivo();
    return 0;
}
```