---------- ERRATA MAPA ----------

Otimização de Rota com Listas Dinâmicas
Uma rede de lojas de produtos naturais utiliza um sistema automatizado para planejar as rotas de entrega dos caminhões. Cada parada da rota representa uma loja que deve ser visitada. No sistema antigo, sempre que uma nova loja era adicionada no meio da rota, era necessário realocar todos os elementos da estrutura, resultando em lentidão e consumo excessivo de memória.
Para resolver esse problema, decidiu-se utilizar uma estrutura de dados dinâmica, onde cada ponto da rota (loja) pudesse ser facilmente inserido ou removido sem a necessidade de deslocar os outros elementos da lista.
Você foi contratado para ajudar a equipe de desenvolvimento a implementar essa nova estrutura. Abaixo está o código inicial de uma lista ligada com algumas funções incompletas. Seu desafio é completar o método inserir_na_posicao para que ele insira uma nova loja (representada por seu nome) em uma posição específica da rota.

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

typedef struct Loja {
    char nome[50];
    struct Loja* proximo;
} Loja;

typedef struct {
    Loja* inicio;
} RotaEntrega;

// Cria nova loja
Loja* criar_loja(const char* nome) {
    Loja* nova = (Loja*)malloc(sizeof(Loja));

    /* função incompleta */

    return nova;
}

// Insere no início da lista
void inserir_inicio(RotaEntrega* rota, const char* nome_loja) {
    Loja* nova_loja = criar_loja(nome_loja);
    nova_loja->proximo = rota->inicio;
    rota->inicio = nova_loja;
}

// Insere na posição específica
void inserir_na_posicao(RotaEntrega* rota, const char* nome_loja, int posicao) {
    if (posicao == 0) {
        inserir_inicio(rota, nome_loja);
```

```
        return;
    }

    Loja* atual = rota->inicio;
    int i = 0;
    while (atual != NULL && i < posicao - 1) {
        atual = atual->proximo;
        i++;
    }

    if (atual == NULL) {
        printf("Posição inválida: %d\n", posicao);
        return;
    }

    /* função incompleta */

}

// Imprime a rota
void imprimir_rota(RotaEntrega* rota) {
    Loja* atual = rota->inicio;
    while (atual != NULL) {
        printf(" %s\n", atual->nome);
        atual = atual->proximo;
    }
}

// Libera memória da rota
void liberar_rota(RotaEntrega* rota) {
    Loja* atual = rota->inicio;
    while (atual != NULL) {
        Loja* temp = atual;
        atual = atual->proximo;
        free(temp);
    }
}

int main() {
    RotaEntrega rota;
    rota.inicio = NULL;

    inserir_inicio(&rota, "Loja D");
    inserir_inicio(&rota, "Loja B");
    inserir_inicio(&rota, "Loja A");

    inserir_na_posicao(&rota, "Loja C", 2);

    printf("Rota final:\n");
    imprimir_rota(&rota);

    liberar_rota(&rota);
    return 0;
```

```
}

# Saída esperada:
# Loja A
# Loja B
# Loja C
# Loja D
```