PROFESSOR: Dr. Marcos Yuzuru
ESTUDANTE: Oscar Borges

Obs: Os métodos requeridos na atividade estão logo antes do main (final do arquivo). Nome dos métodos: listarRecursivo(struct Nodo *nodo) e listarInverso().

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

struct Nodo{
    char dado;
    struct Nodo *ant;
    struct Nodo *prox;
};

struct Nodo *head = NULL;


void inserir_inicio(const char _dado){
    struct Nodo *novo = (struct Nodo*) malloc(sizeof(struct Nodo));
    novo->dado = _dado;

    novo->ant = NULL;
    if (head == NULL) {
        head = novo;
        head->prox = NULL;
    } else {
        head->ant = novo;
        novo->prox = head;
        head = novo;
    }
}

void inserir_fim(const char _dado){
    struct Nodo *novo = (struct Nodo*) malloc(sizeof(struct Nodo));
    novo->dado = _dado;

   if (head == NULL) {
      head = novo;
      head->prox = NULL;
   } else {
       struct Nodo *temp = head;
       while (temp->prox !=  NULL)
          temp = temp->prox;
```

```
        temp->prox = novo;
        novo->ant = temp;
        novo->prox = NULL;
    }
}

void inserir_meio(const char _dado, const int pos){
    struct Nodo *novo = (struct Nodo*) malloc(sizeof(struct Nodo));
    novo->dado = _dado;
    if (pos == 0){
        inserir_inicio(_dado);
    } else {
        struct Nodo *ant_nodo = head;
        struct Nodo *prox_nodo = head;
        int p = 0;
        while ( (p < pos) && (prox_nodo->prox != NULL) ) {
            ant_nodo = prox_nodo;
            prox_nodo = ant_nodo->prox;
            p++;
        }
        prox_nodo->ant = novo;
        ant_nodo->prox = novo;
        novo->ant = ant_nodo;
        novo->prox = prox_nodo;
    }
}

void remover_inicio(){
    if (head != NULL){
      struct Nodo *removido = head;
      head = removido->prox;
      head->ant = NULL;
      free(removido);
      removido = NULL;
    }
}

void remover_meio(const int pos){
    if (head != NULL){
        if (pos == 0){
          remover_inicio();
        } else {
```

```
            int p = 0;
            struct Nodo *prox_nodo = head;
            struct Nodo *ant_nodo = head;

            while ( (p < pos) && (prox_nodo->prox != NULL) ){
              ant_nodo = prox_nodo;
              prox_nodo = ant_nodo->prox;
              p++;
            }

           ant_nodo->prox = prox_nodo->prox;
           prox_nodo->ant = ant_nodo;

           free(prox_nodo);
           prox_nodo = NULL;
         }
    } else {
        printf("Lista vazia!\r\n");
    }
}

void remover_fim(){
    if (head != NULL){
      struct Nodo *ultimo = head;
      struct Nodo *penultimo = NULL;

      while(ultimo->prox != NULL){
          penultimo = ultimo;
          ultimo = penultimo->prox;
      }

      penultimo->prox = NULL;

      free(ultimo);
      ultimo = NULL;
    } else {
        printf("Lista vazia!\r\n");
    }
}

void listar(){
    if (head != NULL){
      struct Nodo *nodo = head;
```

```
      do {
        printf("%c ", nodo->dado);
        nodo = nodo->prox;
      } while (nodo != NULL);
    }else {
      printf("Lista vazia!");
    }
    printf("\r\n");
}

void listarRecursivo(struct Nodo *nodo) {
    if (nodo != NULL) {
        printf("%c ", nodo->dado);
        nodo = nodo->prox;
        listarRecursivo(nodo);
    }else if(head == NULL) {
        printf("Lista vazia!");
    } else {
        printf("\r\n");
    }
}

void listarInverso() {
    if (head != NULL){
      struct Nodo *nodo = head;

      do {
        nodo = nodo->prox;
      } while (nodo->prox != NULL);

      do {
        printf("%c ", nodo->dado);
        nodo = nodo->ant;
      } while (nodo != NULL);

    }else {
      printf("Lista vazia!");
    }
    printf("\r\n");
}

int main(int argc, char **argv)
```

```
{
    listar();
    inserir_inicio('A');
    listar();
    inserir_inicio('B');
    listar();
    inserir_fim('C');
    listar();
    inserir_meio('D',1);
    listar();
    printf("Recursivo \n");
    listarRecursivo(head);
    printf("Inverso \n");
    listarInverso();
    remover_inicio();
    listar();
    remover_meio(1);
    listar();
    remover_fim();
    listar();
    return 0;
}
```