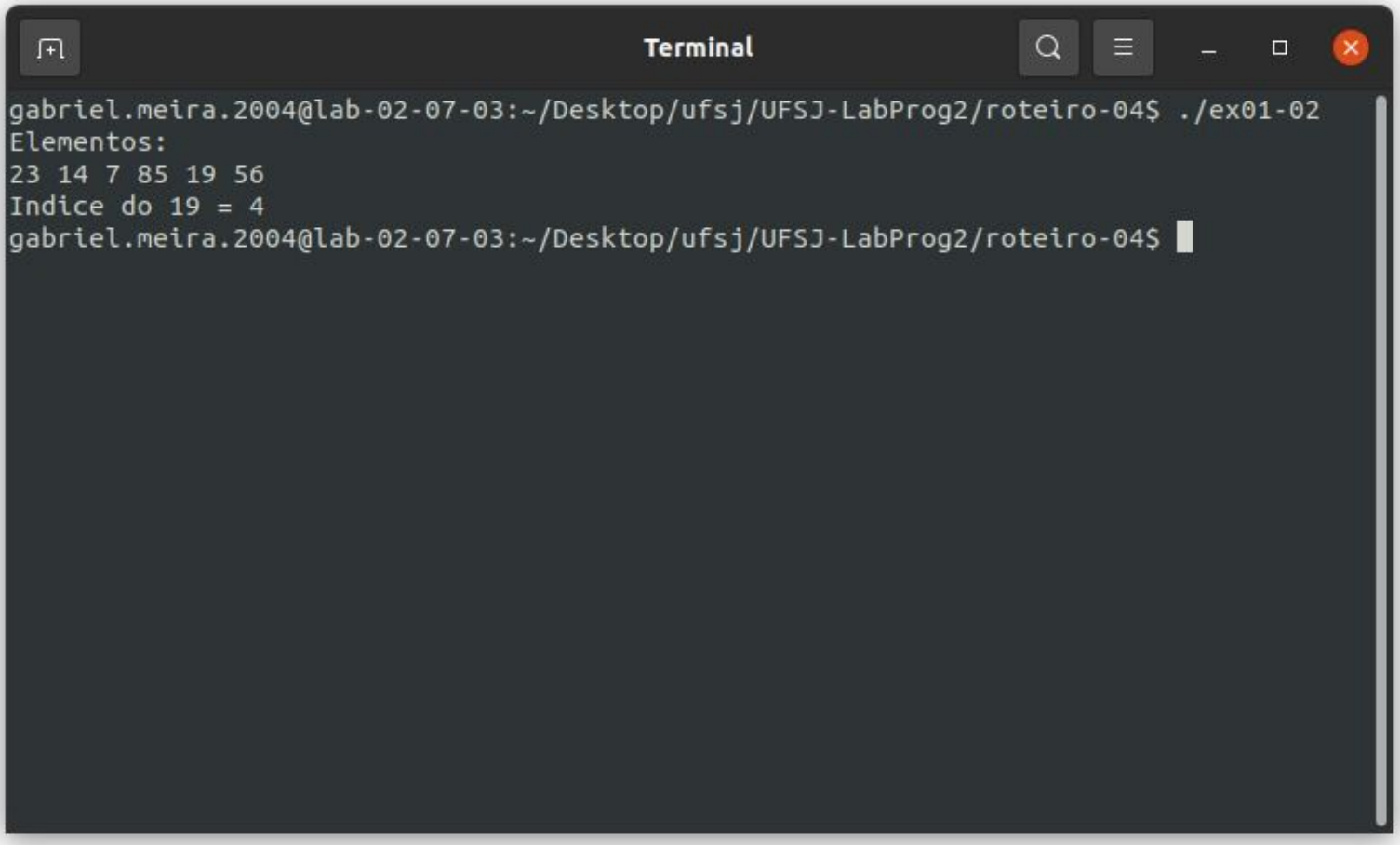
roteiro-04/lista_seq_est.h

```c
/*----------------- File: Lista.h ----------------------+
|Lista Sequencial Estatica                              |
|                                                       |
|                                                       |
| Implementado por Guilherme C. Pena em 12/09/2023      |
+-------------------------------------------------------+ */

#ifndef LISTA_H
#define LISTA_H

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

#define MAX 100

typedef struct {
    int qtd;
    int dados[MAX];
} Lista;

Lista *criaLista() {
    Lista *li;
    li = (Lista *)malloc(sizeof(Lista));
    if (li != NULL)
        li->qtd = 0;
    return li;
}

void destroiLista(Lista *li) {
    if (li != NULL)
        free(li);
}

int tamanhoLista(Lista *li) {
    if (li == NULL)
        return -1;
    return li->qtd;
}

int listaCheia(Lista *li) {
    if (li == NULL)
        return -1;
    return (li->qtd == MAX);
}

int listaVazia(Lista *li) {
    if (li == NULL)
        return -1;
    return (li->qtd == 0);
}

int insereFim(Lista *li, int elem) {
    if (li == NULL) return 0;
    if (!listaCheia(li)) {
        li->dados[li->qtd] = elem;
```

```
56          li->qtd++;
57          return 1;
58      } else {
59          return 0;
60      }
61  }
62
63  int insereIni(Lista *li, int elem) {
64      if (li == NULL) return 0;
65      if (!listaCheia(li)) {
66          int i;
67          for (i = li->qtd; i > 0; i--) {
68              li->dados[i] = li->dados[i - 1];
69          }
70          li->dados[0] = elem;
71          li->qtd++;
72          return 1;
73      } else {
74          return 0;
75      }
76  }
77
78  int imprimeLista(Lista *li) {
79      if (li == NULL) return 0;
80      int i;
81      printf("Elementos:\n");
82      for (i = 0; i < li->qtd; i++) {
83          printf("%d ", li->dados[i]);
84      }
85      printf("\n");
86      return 1;
87  }
88
89  int removeFim(Lista *li) {
90      if (li == NULL) return 0;
91      if (!listaVazia(li)) {
92          li->qtd--;
93          return 1;
94      } else
95          return 0;
96  }
97
98  int removeIni(Lista *li) {
99      if (li == NULL) return 0;
100     if (!listaVazia(li)) {
101         int i;
102         for (i = 0; i < li->qtd - 1; i++)
103             li->dados[i] = li->dados[i + 1];
104         li->qtd--;
105         return 1;
106     } else {
107         return 0;
108     }
109 }
110
111 #endif
```

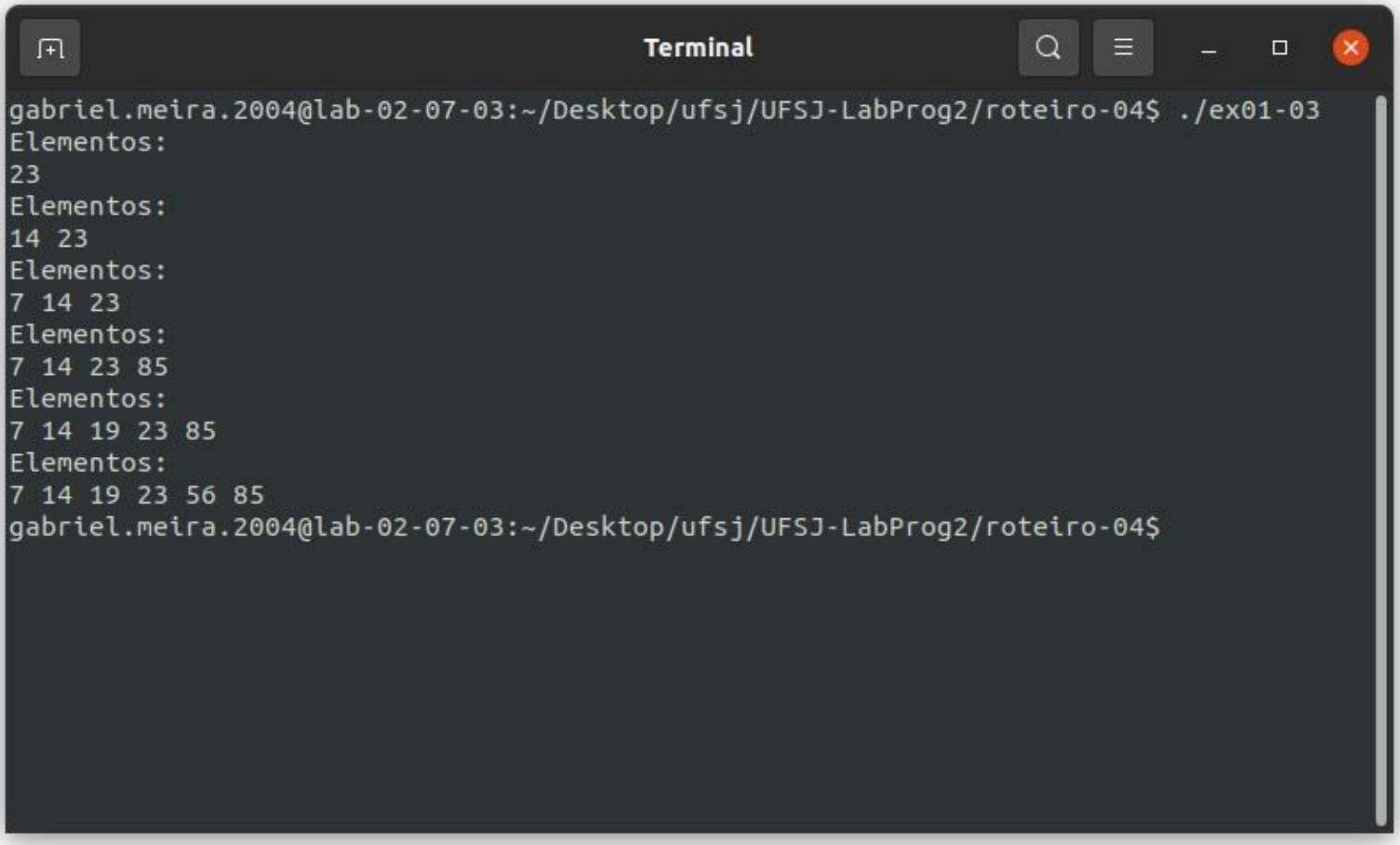
roteiro-04/ex01-02.c

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

#include "lista_seq_est.h"

int procura(Lista* li, int x) {
    for (int i = 0; i < tamanhoLista(li); i++) {
        if (li->dados[i] == x) {
            return i;
        }
    }
    return -1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    printf("Indice do 19 = %d\n", procura(l, 19));

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex01-02
Elementos:
23 14 7 85 19 56
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

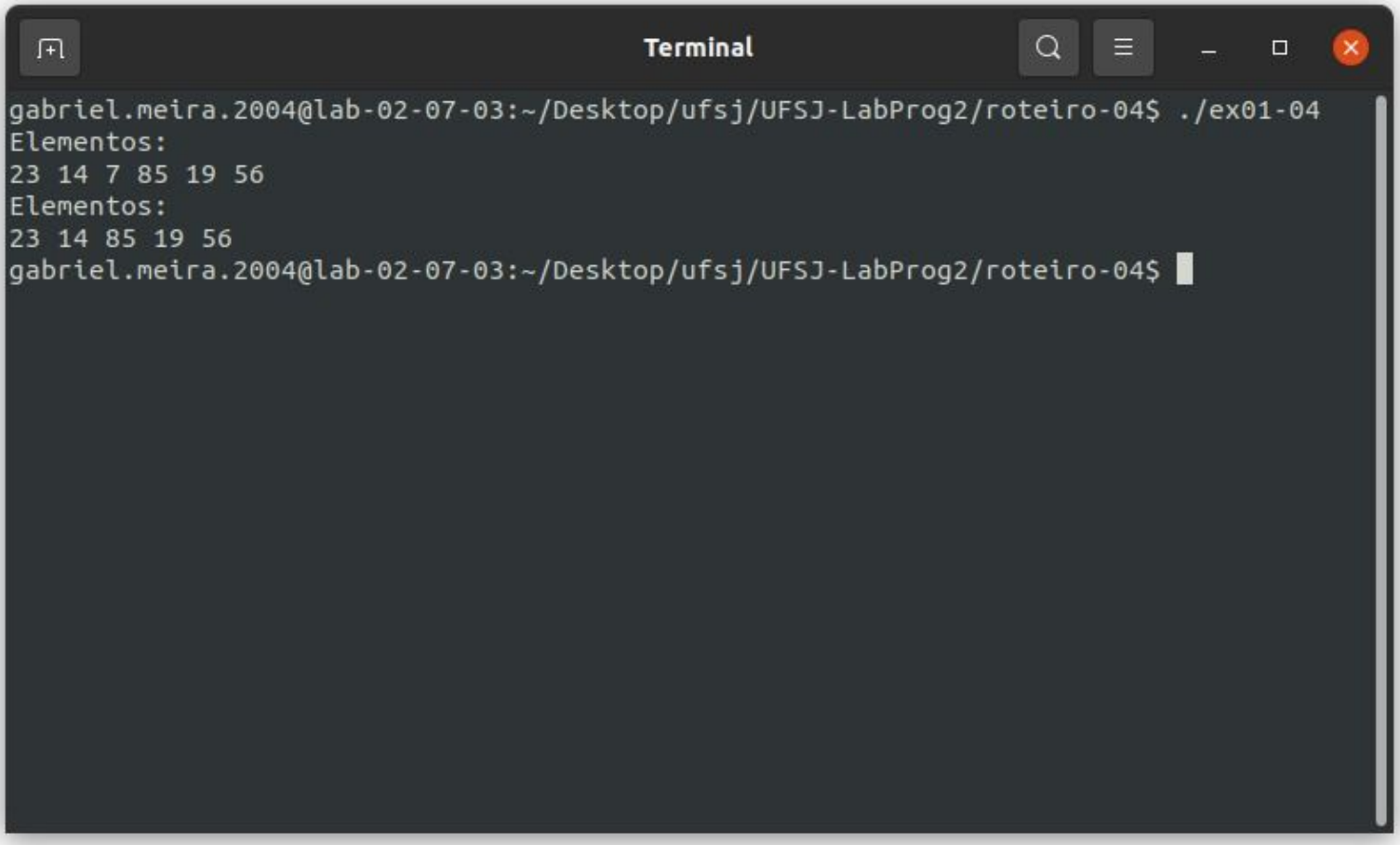
roteiro-04/ex01-03.c

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

#include "lista_seq_est.h"

int insereOrdenado(Lista* li, int x) {
    if (li == NULL || listaCheia(li)) return 0;

    int i;
    for (i = 0; i < tamanhoLista(li) && li->dados[i] < x; i++)
        ;
    for (int j = li->qtd; j > i; j--) {
        li->dados[j] = li->dados[j - 1];
    }
    li->dados[i] = x;
    li->qtd++;

    return 1;
}

int main() {
    Lista* l = criaLista();

    insereOrdenado(l, 23);
    imprimeLista(l);
    insereOrdenado(l, 14);
    imprimeLista(l);
    insereOrdenado(l, 7);
    imprimeLista(l);
    insereOrdenado(l, 85);
    imprimeLista(l);
    insereOrdenado(l, 19);
    imprimeLista(l);
    insereOrdenado(l, 56);
    imprimeLista(l);

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex01-03
Elementos:
23
Elementos:
14 23
Elementos:
7 14 23
Elementos:
7 14 23 85
Elementos:
7 14 19 23 85
Elementos:
7 14 19 23 56 85
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

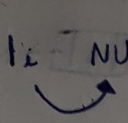
roteiro-04/ex01-04.c

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

#include "lista_seq_est.h"

int removePrimeiroX(Lista* li, int x) {
    if (li == NULL || listaVazia(li)) return 0;

    int i;
    for (i = 0; i < li->qtd; i++) {
        if (li->dados[i] == x) break;
    }

    if (i < li->qtd) {
        for (int j = i; j < li->qtd; j++) {
            li->dados[j] = li->dados[j + 1];
        }
        li->qtd--;
    }

    return 1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    removePrimeiroX(l, 7);
    imprimeLista(l);

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex01-04
Elementos:
23 14 7 85 19 56
Elementos:
23 14 85 19 56
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

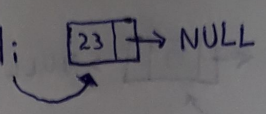
2.1) Lista Simplesmente Encadeada

I) Lista vazia



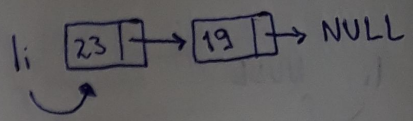
- criar ponteiro "li" para NULL

II) Inserção (23)



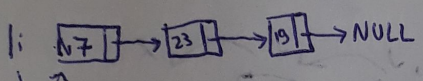
- Criar novo nó (23)
- "li" aponta para o novo nó
- o nó aponta para NULL

III) Inserção Fim (19)



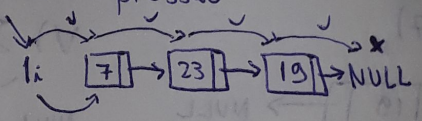
- criar novo nó (19)
- "23" que apontava para NULL vai apontar para "19"
- "19" vai apontar para NULL

IV) Inserção Início



- Criar novo nó (7)
- "7" vai apontar para o nó que "li" aponta (23)
- "li" agora aponta para "7"

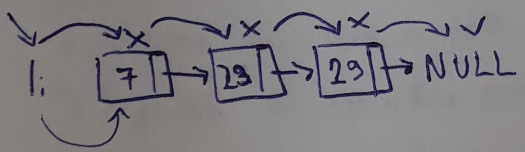
V) Impressão



$\Rightarrow \{ 7, 23, 19 \}$

- se a lista for vazia não imprime
- caso tenha elementos percorre todos os nós até encontrar NULL.

VI) Liberação da Lista



{ Del: 7, Del: 23, Del 19 }

- caso a lista esteja vazia, libera a lista
- caso haja elementos, percorre de 1 em 1 cada nó, desalocando a memória, até que chegue em NULL, executando o passo anterior

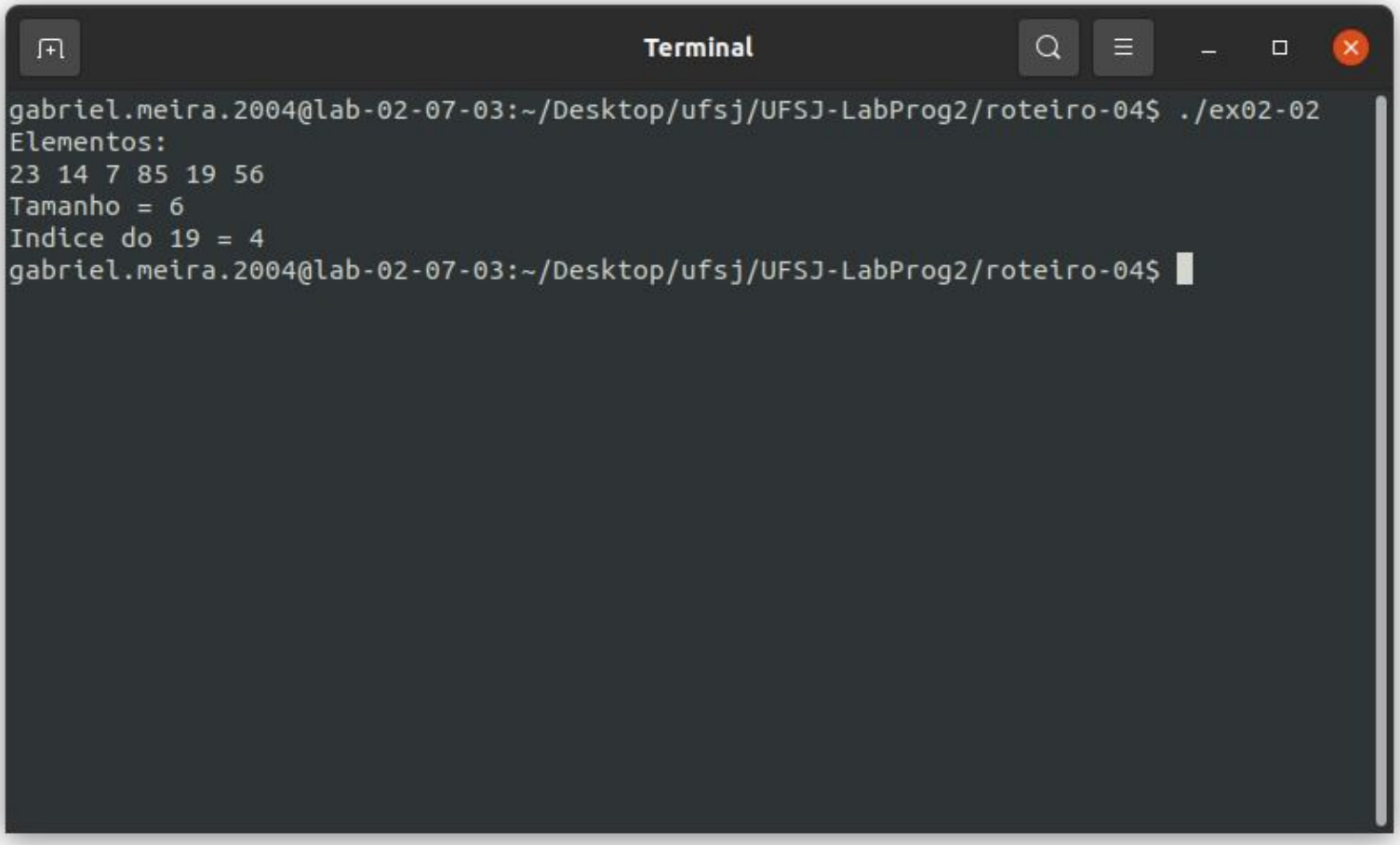
roteiro-04/lista_sim_enc.h

```c
/*----------------- File: LSE.h  ----------------------+
|Lista Simplesmente Encadeada                          |
| Implementado por Guilherme C. Pena em 14/09/2023     |
+------------------------------------------------------+ */
#ifndef LISTASE_H
#define LISTASE_H

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

typedef struct NO {
    int info;
    struct NO* prox;
} NO;
typedef struct NO* Lista;

Lista* criaLista() {
    Lista* li;
    li = (Lista*)malloc(sizeof(Lista));
    if (li != NULL) {
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista* li) {
    if (li == NULL) return 1;
    if (*li == NULL) return 1;  // True - Vazia!
    return 0;                   // False - tem elemento!
}

NO* alocarNO() {
    return (NO*)malloc(sizeof(NO));
}

void liberarNO(NO* q) {
    free(q);
}

int insereIni(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    novo->prox = *li;
    *li = novo;
    return 1;
}

int insereFim(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if (listaVazia(li)) {
        *li = novo;
    } else {
```

```
59          NO* aux = *li;
60          while (aux->prox != NULL)
61              aux = aux->prox;
62          aux->prox = novo;
63      }
64      return 1;
65  }
66
67  int removeIni(Lista* li) {
68      if (li == NULL) return 0;
69      if (listaVazia(li)) return 0;
70      NO* aux = *li;
71      *li = aux->prox;
72      liberarNO(aux);
73      return 1;
74  }
75
76  int removeFim(Lista* li) {
77      if (li == NULL) return 0;
78      if (listaVazia(li)) return 0;
79      NO *ant, *aux = *li;
80      while (aux->prox != NULL) {
81          ant = aux;
82          aux = aux->prox;
83      }
84      if (aux == *li)
85          *li = aux->prox;
86      else
87          ant->prox = aux->prox;
88      liberarNO(aux);
89      return 1;
90  }
91
92  void imprimeLista(Lista* li) {
93      if (li == NULL) return;
94      if (listaVazia(li)) {
95          printf("Lista Vazia!\n");
96          return;
97      }
98      printf("Elementos:\n");
99      NO* aux = *li;
100     while (aux != NULL) {
101         printf("%d ", aux->info);
102         aux = aux->prox;
103     }
104     printf("\n");
105 }
106
107 void destroiLista(Lista* li) {
108     if (li != NULL) {
109         NO* aux;
110         while ((*li) != NULL) {
111             aux = *li;
112             *li = (*li)->prox;
113             liberarNO(aux);
114         }
115         free(li);
116     }
117 }
118
119 #endif
```

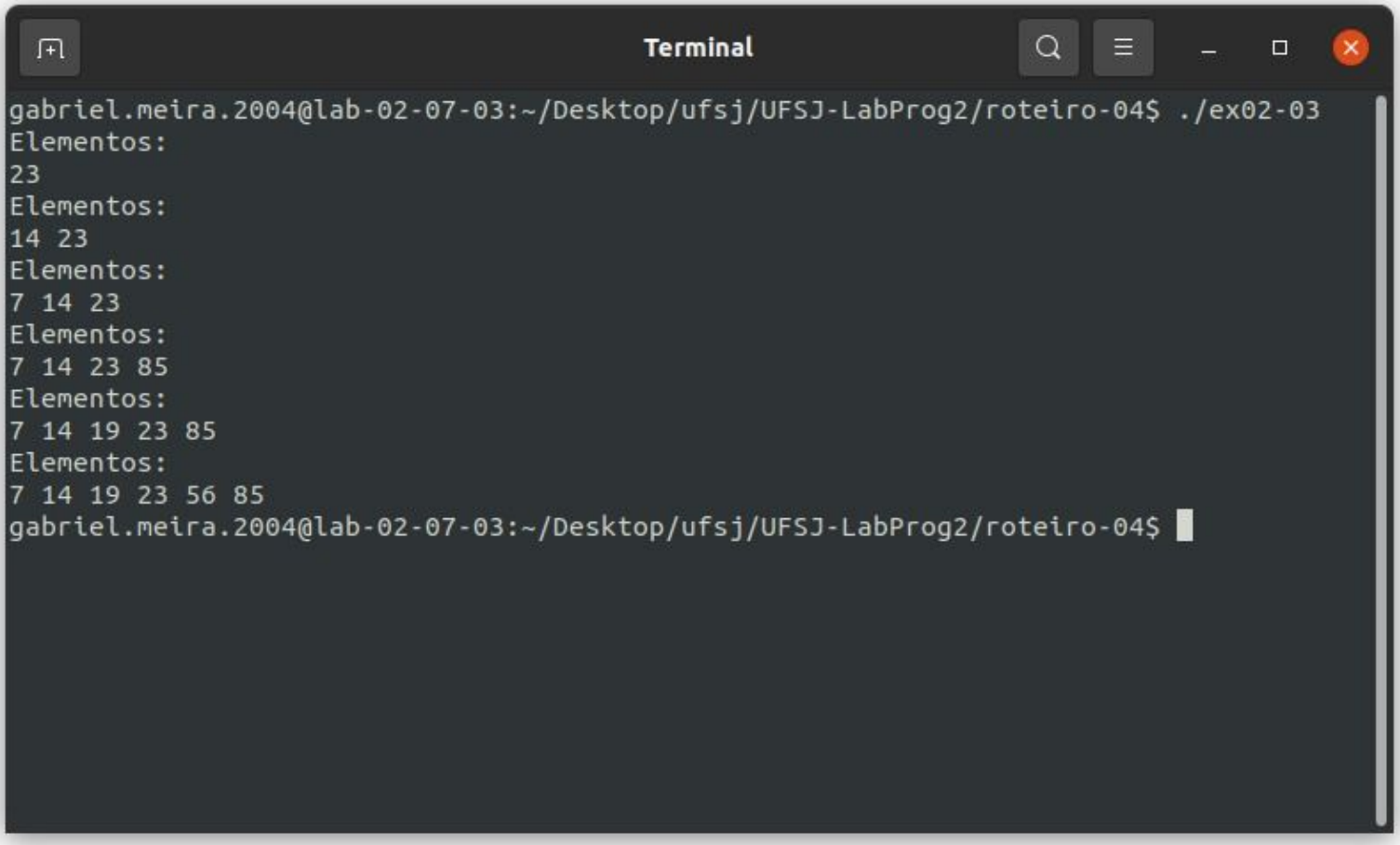
roteiro-04/ex02-02.c

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

#include "lista_sim_enc.h"

int tamanho(Lista* li) {
    int t = 0;
    NO* p = *li;
    while (p != NULL) {
        p = p->prox;
        t++;
    }

    return t;
}

int procura(Lista* li, int x) {
    int i = 0;
    NO* p = *li;
    while (p != NULL) {
        if (p->info == x) return i;
        p = p->prox;
        i++;
    }

    return -1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    printf("Tamanho = %d\n", tamanho(l));
    printf("Indice do 19 = %d\n", procura(l, 19));

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex02-02
Elementos:
23 14 7 85 19 56
Tamanho = 6
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

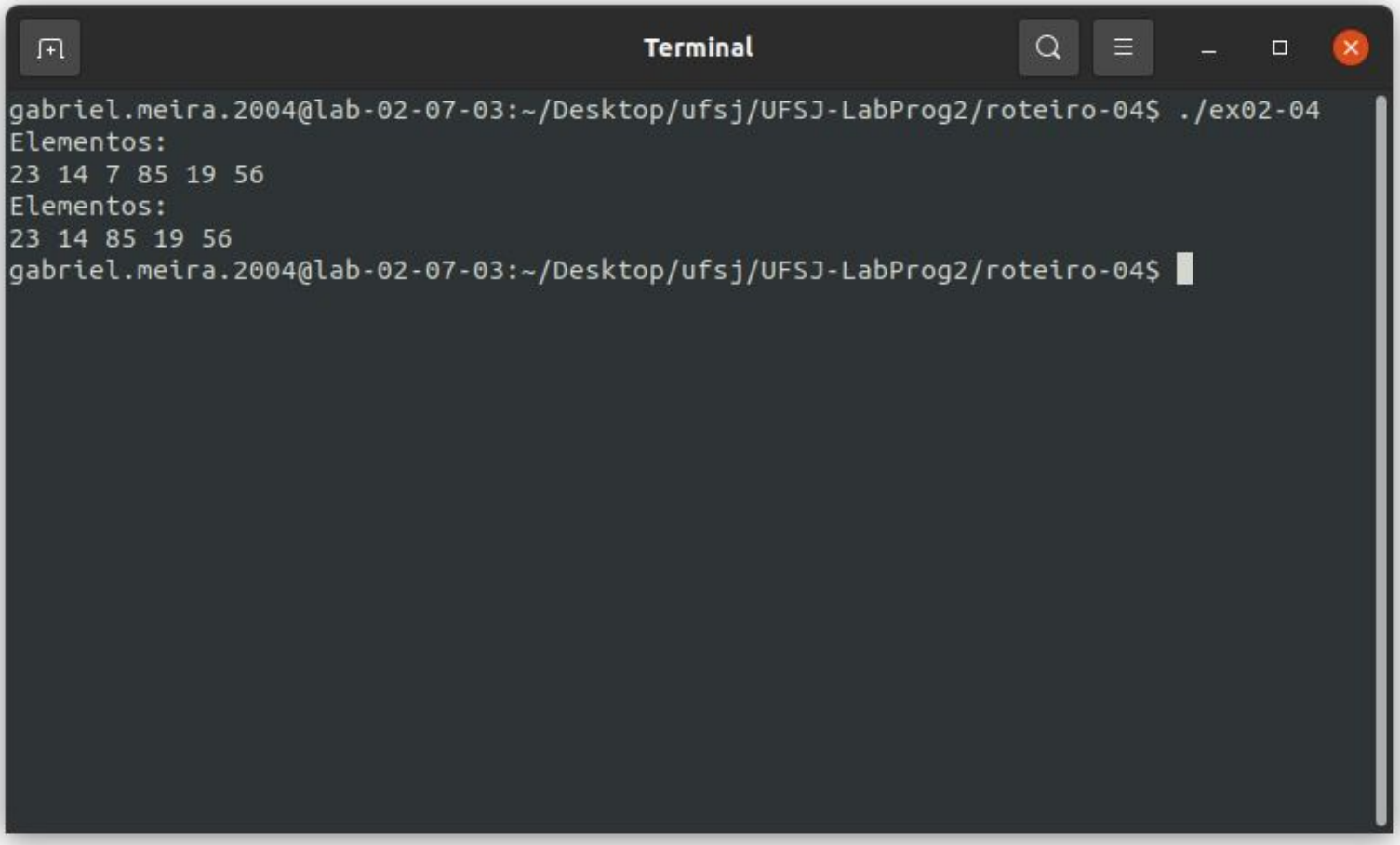
roteiro-04/ex02-03.c

```c
int insereOrdenado(Lista* li, int x) {
    if (li == NULL) return 0;

    NO* novo = alocarNO();
    novo->info = x;
    novo->prox = NULL;

    if (listaVazia(li)) {
        *li = novo;
    } else {
        NO i;
        i.prox = *li;

        NO* p = &i;
        while (p->prox != NULL && p->prox->info < x) {
            p = p->prox;
        }

        if (p->prox == *li) {
            novo->prox = *li;
            *li = novo;
        }

        novo->prox = p->prox;
        p->prox = novo;
    }

    return 1;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex02-03
Elementos:
23
Elementos:
14 23
Elementos:
7 14 23
Elementos:
7 14 23 85
Elementos:
7 14 19 23 85
Elementos:
7 14 19 23 56 85
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

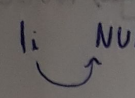
roteiro-04/ex02-04.c

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

#include "lista_sim_enc.h"

int removePrimeiroX(Lista* li, int x) {
    if (li == NULL || listaVazia(li)) return 0;

    NO i;  // No inicial auxiliar
    i.prox = *li;

    NO* p = &i;
    while (p->prox != NULL) {
        if (p->prox->info == x) {
            NO* aux = p->prox;
            p->prox = aux->prox;
            liberarNO(aux);
            break;
        }

        p = p->prox;
    }

    return 1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    removePrimeiroX(l, 7);
    imprimeLista(l);

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex02-04
Elementos:
23 14 7 85 19 56
Elementos:
23 14 85 19 56
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

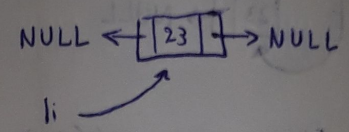
3.1) Lista Duplamente Encadeada

I) Lista vazia



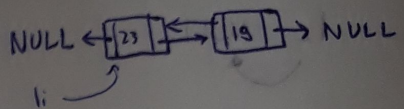
- Cria ponteiro "li" para NULL

II) Inserção (23)



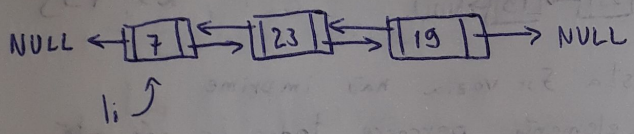
- Cria novo nó (23)
- anterior e próximo são NULL
- "li" aponta para "23"

III) Inserção Fim (19)



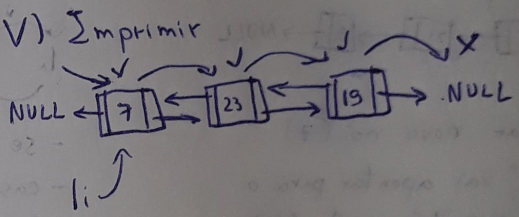
- Cria novo nó (19)
- "23" deixa de apontar para o próximo NULL e aponta para "19", cujo próximo é NULL
- O anterior de "19" é "23"

IV) Inserção Início (7)



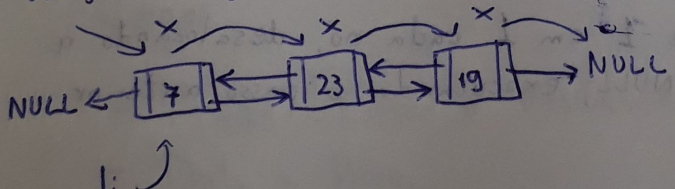
- Cria novo nó (7)
- o anterior de "7" é NULL e o próximo é o nó apontado por "li"
- o anterior de "li" será o "7"
- "li" agora será o "7"

V) Imprimir



- caso a lista seja vazia não imprime
- percorre todos os nós imprindo seus dados até encontrar NULL

VI) Liberação da lista



{ Del: 7, Del: 23, Del: 19}

- caso a lista esteja vazia, libera a lista
- caso haja elementos, percorre a lista liberando cada nó até chegar em NULL

roteiro-04/lista_dup_enc.h

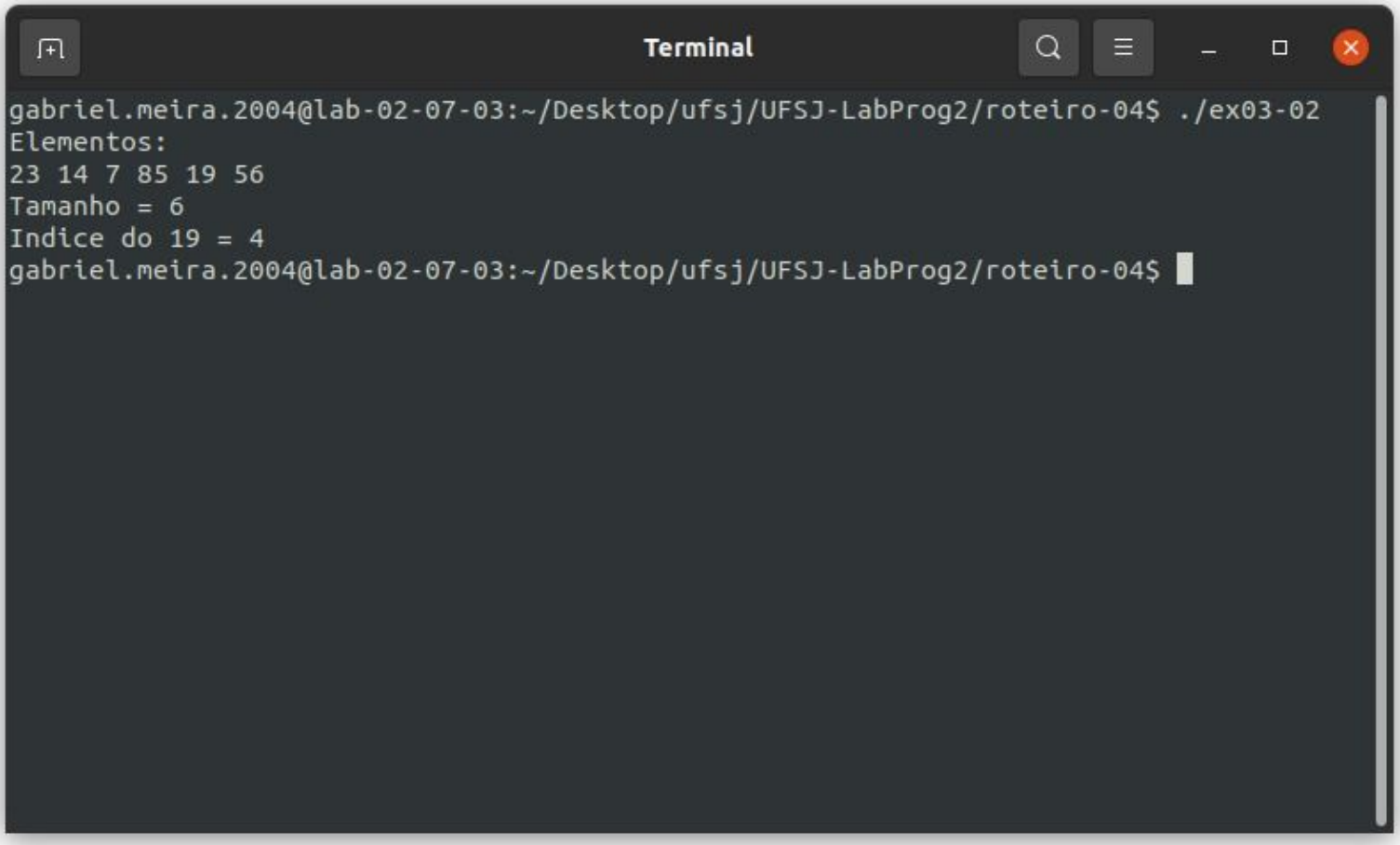
```c
/*----------------- File: LDE.h  ----------------------+
|Lista Duplamente Encadeada                           |
|                                                     |
|                                                     |
| Implementado por Guilherme C. Pena em 19/09/2023    |
+-----------------------------------------------------+ */

#ifndef LDE_H
#define LDE_H

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

typedef struct NO {
    int info;
    struct NO* prox;
    struct NO* ant;
} NO;

typedef struct NO* Lista;

Lista* criaLista() {
    Lista* li;
    li = (Lista*)malloc(sizeof(Lista));
    if (li != NULL) {
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista* li) {
    if (li == NULL) return 1;
    if (*li == NULL) return 1;  // True - Vazia!
    return 0;                   // False - tem elemento!
}

NO* alocarNO() {
    return (NO*)malloc(sizeof(NO));
}

void liberarNO(NO* q) {
    free(q);
}

int insereIni(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    novo->prox = *li;
    novo->ant = NULL;
    if (!listaVazia(li))
        (*li)->ant = novo;
    *li = novo;
    return 1;
```

```c
}

int insereFim(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if (listaVazia(li)) {
        novo->ant = NULL;
        *li = novo;
    } else {
        NO* aux = *li;
        while (aux->prox != NULL)
            aux = aux->prox;
        aux->prox = novo;
        novo->ant = aux;
    }
    return 1;
}

int removeIni(Lista* li) {
    if (li == NULL) return 0;
    if (listaVazia(li)) return 0;
    NO* aux = *li;
    *li = aux->prox;
    if (aux->prox != NULL)
        aux->prox->ant = NULL;
    liberarNO(aux);
    return 1;
}

int removeFim(Lista* li) {
    if (li == NULL) return 0;
    if (listaVazia(li)) return 0;
    NO* aux = *li;
    while (aux->prox != NULL)
        aux = aux->prox;
    if (aux->ant == NULL)
        *li = aux->prox;
    else
        aux->ant->prox = NULL;
    liberarNO(aux);
    return 1;
}

void imprimeLista(Lista* li) {
    if (li == NULL) return;
    if (listaVazia(li)) {
        printf("Lista Vazia!\n");
        return;
    }
    printf("Elementos:\n");
    NO* aux = *li;
    while (aux != NULL) {
        printf("%d ", aux->info);
        aux = aux->prox;
    }
    printf("\n");
```

```
}

void destroiLista(Lista* li) {
    if (li != NULL) {
        NO* aux;
        while ((*li) != NULL) {
            aux = *li;
            *li = (*li)->prox;
            // printf("Destruindo.. %d\n", aux->info);
            liberarNO(aux);
        }
        free(li);
    }
}

#endif
```

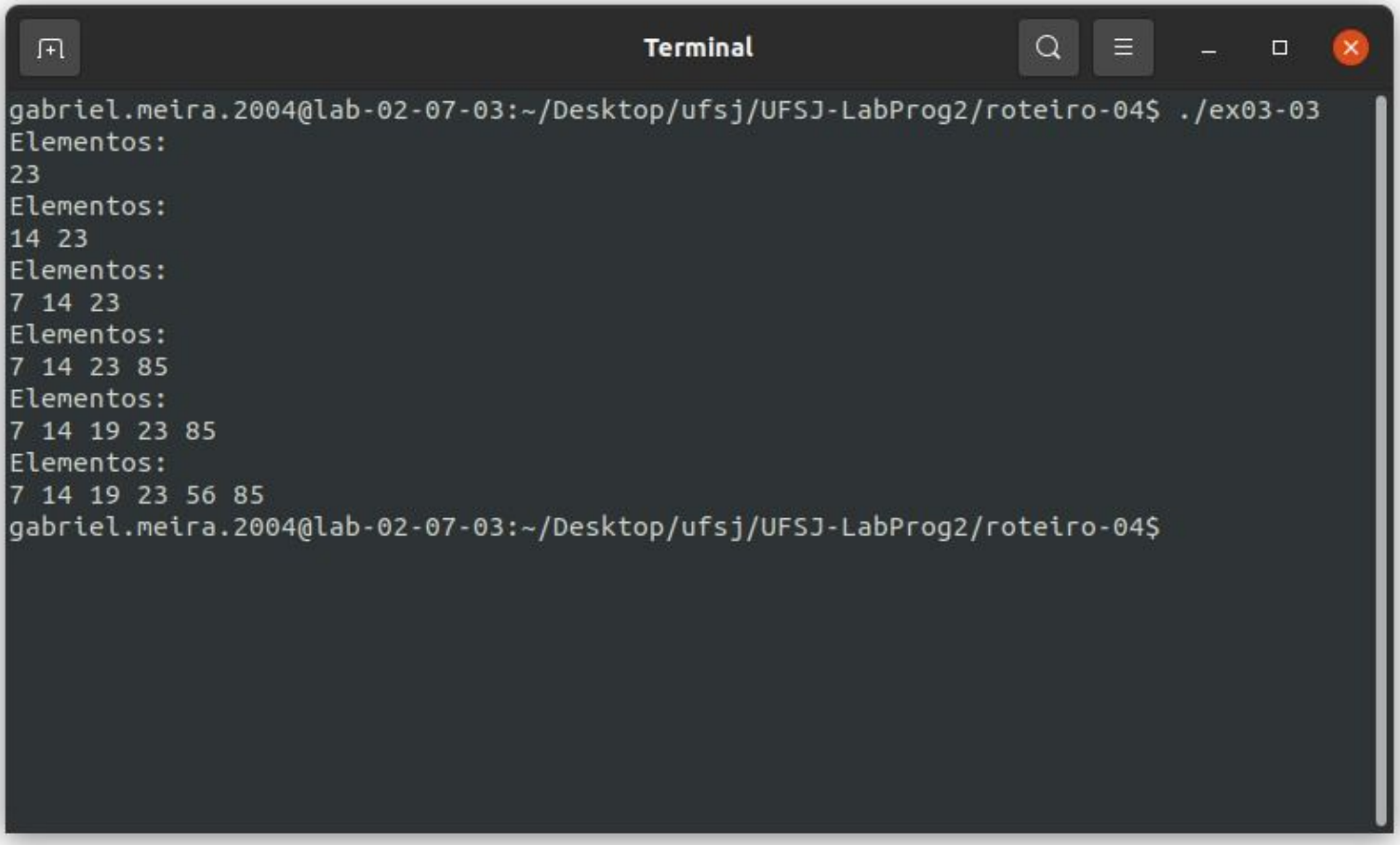
roteiro-04/ex03-02.c

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

#include "lista_dup_enc.h"

int tamanho(Lista* li) {
    int t = 0;
    NO* p = *li;
    while (p != NULL) {
        p = p->prox;
        t++;
    }

    return t;
}

int procura(Lista* li, int x) {
    int i = 0;

    NO* p = *li;
    while (p != NULL) {
        if (p->info == x) return i;
        p = p->prox;
        i++;
    }

    return -1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    printf("Tamanho = %d\n", tamanho(l));
    printf("Indice do 19 = %d\n", procura(l, 19));

    destroiLista(l);

    return 0;
}

```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex03-02
Elementos:
23 14 7 85 19 56
Tamanho = 6
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

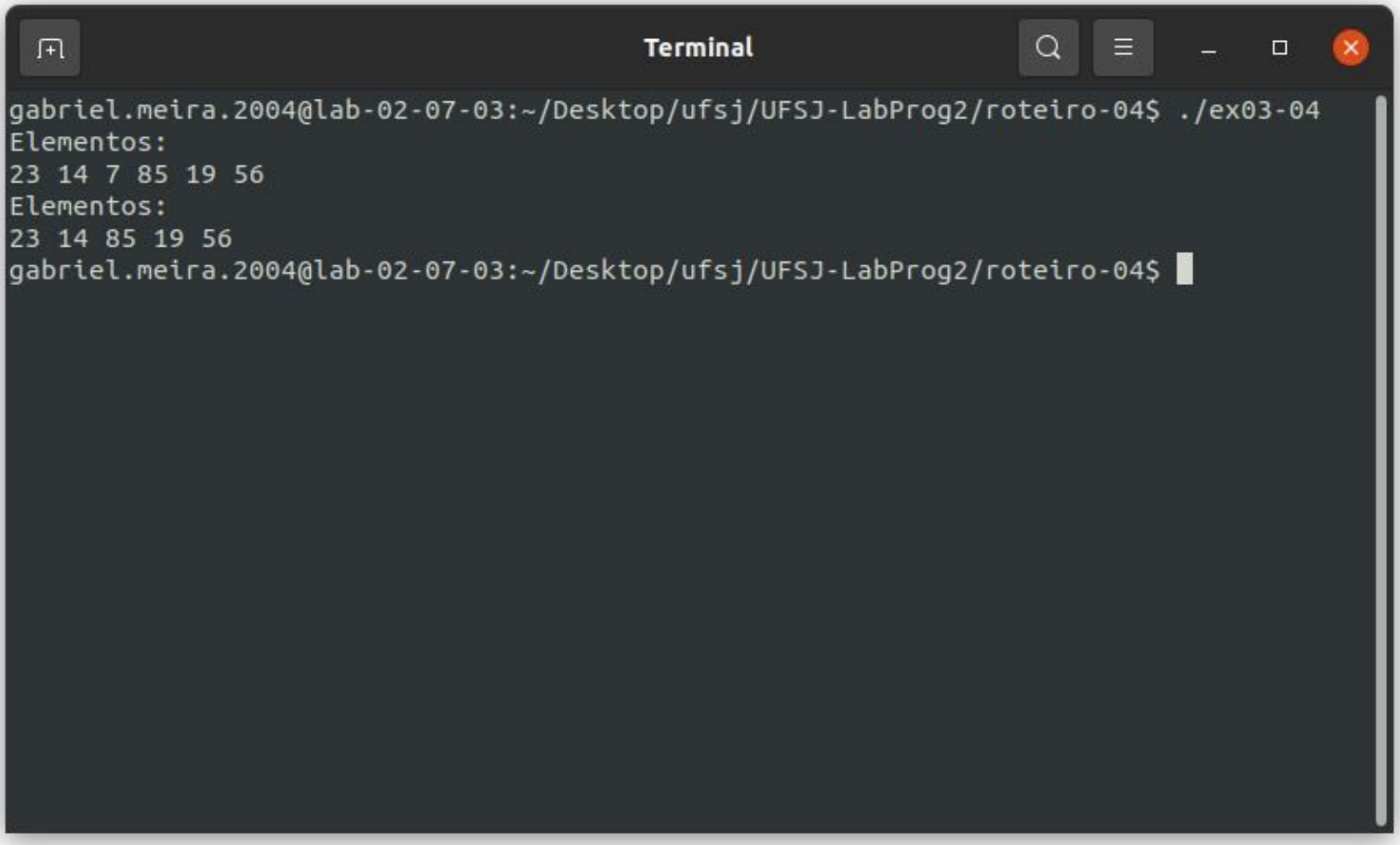
roteiro-04/ex03-03.c

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

#include "lista_dup_enc.h"

int insereOrdenado(Lista* li, int x) {
    if (li == NULL) return 0;

    NO* novo = alocarNO();
    novo->info = x;
    novo->ant = NULL;
    novo->prox = NULL;

    if (listaVazia(li)) {
        *li = novo;
    } else {
        NO i;
        i.prox = *li;

        NO* p = &i;
        while (p->prox != NULL && p->prox->info < x) {
            p = p->prox;
        }

        if (p->prox == *li) {
            novo->prox = *li;
            *li = novo;
        } else {
            novo->ant = p;
        }
        novo->prox = p->prox;
        p->prox = novo;
    }

    return 1;
}

int main() {
    Lista* l = criaLista();

    insereOrdenado(l, 23);
    imprimeLista(l);
    insereOrdenado(l, 14);
    imprimeLista(l);
    insereOrdenado(l, 7);
    imprimeLista(l);
    insereOrdenado(l, 85);
    imprimeLista(l);
    insereOrdenado(l, 19);
    imprimeLista(l);
    insereOrdenado(l, 56);
    imprimeLista(l);

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex03-03
Elementos:
23
Elementos:
14 23
Elementos:
7 14 23
Elementos:
7 14 23 85
Elementos:
7 14 19 23 85
Elementos:
7 14 19 23 56 85
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

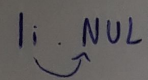
roteiro-04/ex03-04.c

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

#include "lista_dup_enc.h"

int removePrimeiroX(Lista* li, int x) {
    if (li == NULL || listaVazia(li)) return 0;

    NO i;  // No inicial auxiliar
    i.prox = *li;

    NO* p = &i;
    while (p->prox != NULL) {
        if (p->prox->info == x) {
            NO* aux = p->prox;
            p->prox = aux->prox;
            p->prox->ant = aux->ant;
            liberarNO(aux);
            break;
        }

        p = p->prox;
    }

    return 1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    removePrimeiroX(l, 7);
    imprimeLista(l);

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex03-04
Elementos:
23 14 7 85 19 56
Elementos:
23 14 85 19 56
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$

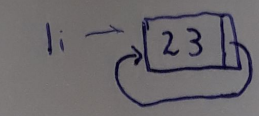
4.1) Lista Circular Simplesmente Encadeada

I) Lista vazia



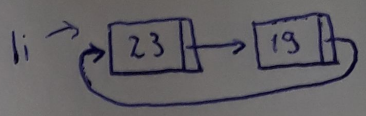
- Cria o ponteiro "li" para NULL

II) Inserção (23)



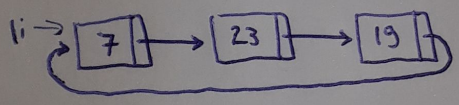
- Cria o novo nó (23)
- o próximo de "23" é ele mesmo
- "li" aponta para "23"

III) Inserção Fim (19)



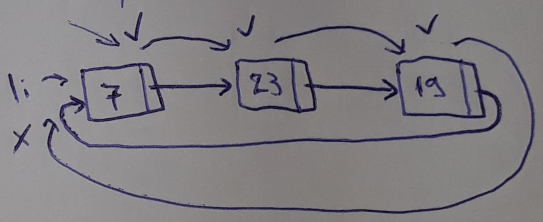
- Cria o novo nó (19)
- o próximo de "19" é o nó apontado por "li"
- o último nó, que aponta para "li" passa a apontar para o "19".

IV) Inserção Início (7)



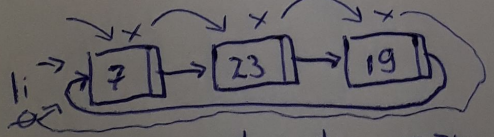
- Cria o novo nó (7)
- o próximo de "7" será "li"
- o último nó, que aponta para "li" passa a apontar para "7"
- "li" aponta para "7"

V) Imprimir



- Caso a lista esteja vazia não imprime
- percorre a lista imprimindo os dados de cada nó até que "li" seja visitado pela segunda vez (fim da lista)

VI) Liberar Lista



{Del: 7, Del: 23, Del: 19}

- Caso a lista esteja vazia, libera a lista
- Caso haja elementos, percorre a lista deletando cada nó até que "li" seja visitado pela segunda vez

roteiro-04/lista_cir_sim_enc.h

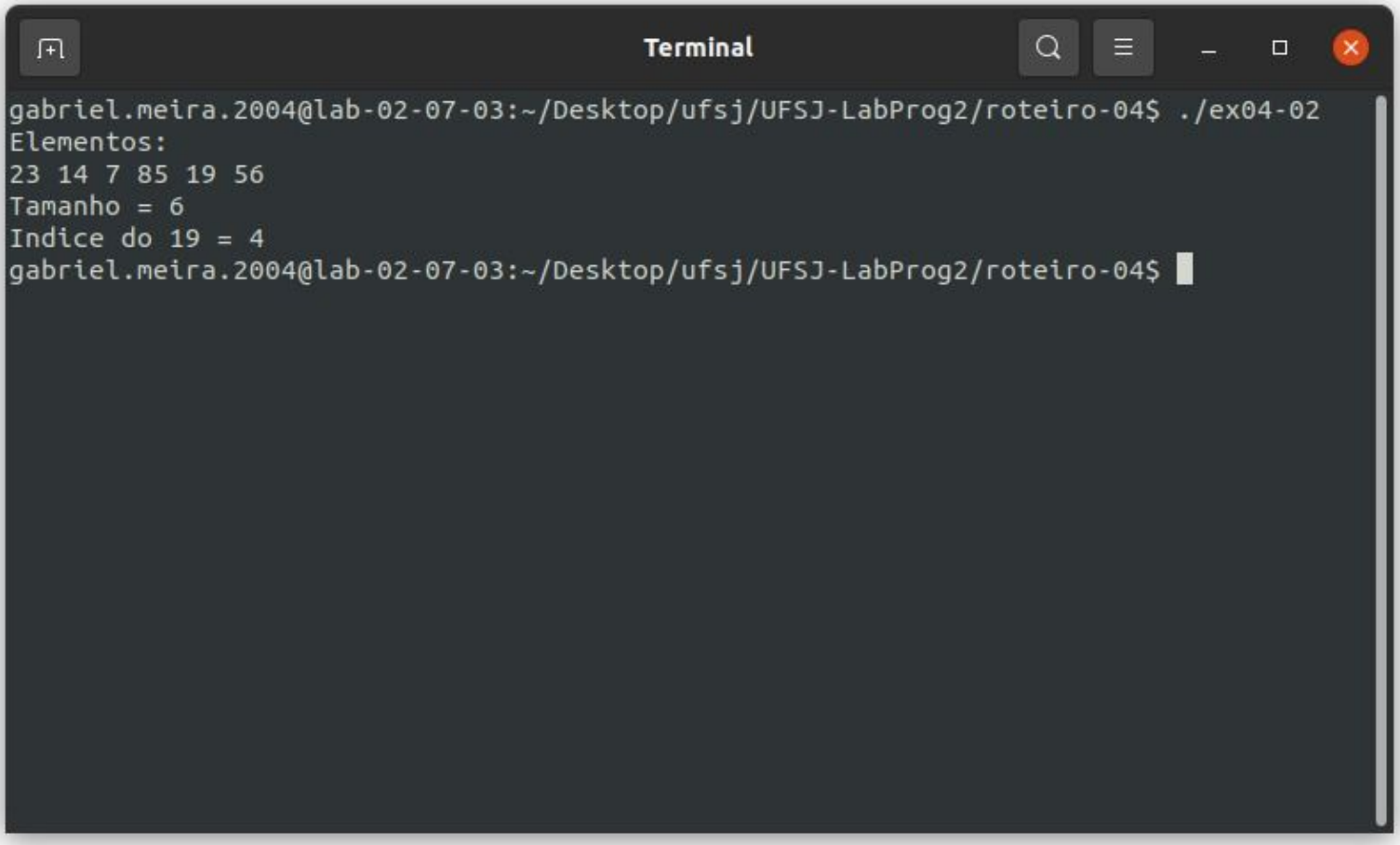
```c
/*----------------- File: LCSE.h  ----------------------+
|Lista Circular Simplesmente Encadeada                  |
|                                                       |
|                                                       |
| Implementado por Guilherme C. Pena em 19/09/2023      |
+-------------------------------------------------------+ */

#ifndef LCSE_H
#define LCSE_H

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

typedef struct NO {
    int info;
    struct NO* prox;
} NO;

typedef struct NO* Lista;

Lista* criaLista() {
    Lista* li;
    li = (Lista*)malloc(sizeof(Lista));
    if (li != NULL) {
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista* li) {
    if (li == NULL) return 1;
    if (*li == NULL) return 1;  // True - Vazia!
    return 0;                   // False - tem elemento!
}

NO* alocarNO() {
    return (NO*)malloc(sizeof(NO));
}

void liberarNO(NO* q) {
    free(q);
}

int insereIni(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    if (listaVazia(li)) {
        novo->prox = novo;
        *li = novo;
    } else {
        NO* aux = *li;
        while (aux->prox != (*li))
            aux = aux->prox;
```

```
56          aux->prox = novo;
57          novo->prox = *li;
58          *li = novo;
59      }
60      return 1;
61  }
62
63  int insereFim(Lista* li, int elem) {
64      if (li == NULL) return 0;
65      NO* novo = alocarNO();
66      if (novo == NULL) return 0;
67      novo->info = elem;
68      if (listaVazia(li)) {
69          novo->prox = novo;
70          *li = novo;
71      } else {
72          NO* aux = *li;
73          while (aux->prox != (*li))
74              aux = aux->prox;
75          aux->prox = novo;
76          novo->prox = *li;
77      }
78      return 1;
79  }
80
81  int removeIni(Lista* li) {
82      if (li == NULL) return 0;
83      if (listaVazia(li)) return 0;
84      NO* prim = *li;
85      if (prim == prim->prox) {
86          // apenas 1 elemento
87          *li = NULL;
88      } else {
89          NO* aux = *li;
90          while (aux->prox != (*li))
91              aux = aux->prox;
92          aux->prox = (*li)->prox;
93          *li = (*li)->prox;
94      }
95      liberarNO(prim);
96      return 1;
97  }
98
99  int removeFim(Lista* li) {
100     if (li == NULL) return 0;
101     if (listaVazia(li)) return 0;
102     NO* aux = *li;
103     if (aux == aux->prox) {
104         // apenas 1 elemento
105         *li = NULL;
106     } else {
107         NO* ant;
108         while (aux->prox != (*li)) {
109             ant = aux;  // anterior
110             aux = aux->prox;
111         }
112         ant->prox = *li;
113     }
114     liberarNO(aux);
```

```c
    return 1;
}

void imprimeLista(Lista* li) {
    if (li == NULL) return;
    if (listaVazia(li)) {
        printf("Lista Vazia!\n");
        return;
    }
    printf("Elementos:\n");
    NO* aux = *li;
    while (aux->prox != *li) {
        printf("%d ", aux->info);
        aux = aux->prox;
    }
    printf("%d ", aux->info);
    printf("\n");
}

void destroiLista(Lista* li) {
    if (li != NULL && (*li) != NULL) {
        NO *prim, *aux;
        prim = *li;
        *li = (*li)->prox;
        while ((*li) != prim) {
            aux = *li;
            *li = (*li)->prox;
            liberarNO(aux);
        }
        liberarNO(prim);
        free(li);
    }
}

#endif
```

roteiro-04/ex04-02.c

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

#include "lista_cir_sim_enc.h"

int tamanho(Lista* li) {
    int t = 0;
    NO* p = *li;

    while (p != *li || t == 0) {
        p = p->prox;
        t++;
    }

    return t;
}

int procura(Lista* li, int x) {
    int i = 0;
    NO* p = *li;
    while (p != *li || i == 0) {
        if (p->info == x) return i;
        p = p->prox;
        i++;
    }

    return -1;
}

int main() {
    Lista* l = criaLista();

    insereFim(l, 23);
    insereFim(l, 14);
    insereFim(l, 7);
    insereFim(l, 85);
    insereFim(l, 19);
    insereFim(l, 56);

    imprimeLista(l);
    printf("Tamanho = %d\n", tamanho(l));
    printf("Indice do 19 = %d\n", procura(l, 19));

    destroiLista(l);

    return 0;
}
```


Terminal
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex04-02
Elementos:
23 14 7 85 19 56
Tamanho = 6
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$