Nome: Rodrigo Silva de Oliveira **RA:** 1840482113026 **ADS – NOITE**

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

void exibe (Lista L) {
	while (L != NULL) {
		printf("%d\n\n", L ->
item);
		L = L -> prox;
	}
}

int main(void) {
	printf(" **********
EXERCÍCIO 1. PROGRAMA PARA
CRIAÇÃO E EXIBIÇÃO
**********\n\n", setlocale(LC_ALL,
""));

	Lista l = no(3, no(1, no(5,
NULL)));
	exibe(l);

	printf("\n");
	system ("pause");
	return 0;
}
```

```
********** EXERCÍCIO 1. PROGRAMA PARA CRIAÇÃO E EXI

       3

       1

       5

Pressione qualquer tecla para continuar. . .
```

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

void exibe (Lista L) {
	printf("\t[");
	while (L != NULL) {
		if (L -> prox != NULL){
			printf("%d", L ->
item);
				printf(" ,
");
			L = L -> prox;
	} else {
			printf("%d", L ->
item);
			L = L -> prox;
		}
}
	printf("]");
}

int main(void) {
	printf(" **********
EXERCÍCIO 2. OUTRA FORMA DE
EXIBIÇÃO **********\n\n",
setlocale(LC_ALL, ""));

	Lista I = no(3, no(1, no(5,
NULL)));
	exibe(I);

	printf("\n\n");
```

```
********** EXERCÍCIO 2. OUTRA FORMA DE EXIBIÇÃO *

	[3 , 1 , 5]

Pressione qualquer tecla para continuar. . .
```

```
        system ("pause");
        return 0;
}
```

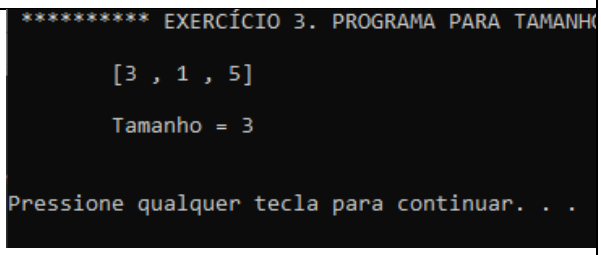
```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

typedef int Item;
typedef struct no {
        Item item;
        struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
        Lista n = malloc(sizeof(struct
no));
        n -> item = x;
        n -> prox = p;
        return n;
}

int tamanho (Lista L) {
        int t = 0;
        while (L) {
                t++;
                L = L -> prox;
        }
        return t;
}

void exibe (Lista L) {
        printf("\t[");
        while (L != NULL) {
                if (L -> prox != NULL){
                        printf("%d", L ->
item);
                                printf(" ,
");
                        L = L -> prox;
        } else {
                        printf("%d", L ->
item);
                        L = L -> prox;
                }
}
        printf("]");
}
```



```
int main(void) {
	printf(" **********
EXERCÍCIO 3. PROGRAMA PARA
TAMANHO **********\n\n",
setlocale(LC_ALL, ""));

	Lista I = no(3, no(1, no(5,
NULL)));
	exibe(I);

	printf("\n\n\tTamanho =
%d\n", tamanho (I));
	printf("\n\n");
	system ("pause");
	return 0;
}
```

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

int tamanho (Lista L) {
	int t = 0;
	while (L) {
		t++;
		L = L -> prox;
	}
	return t;
}

int soma (Lista L) {
	int somatoria = 0;
	while (L != NULL) {
```

```
********** EXERCÍCIO 4. SOMA DOS ITENS DA L

       [3 , 1 , 5]

       Tamanho = 3

       Soma dos itens = 9

Pressione qualquer tecla para continuar. .
```

```
                somatoria = somatoria
+ L -> item;
                L = L -> prox;
        }
        return somatoria;
}

void exibe (Lista L) {
        printf("\t[");
        while (L != NULL) {
                if (L -> prox != NULL){
                        printf("%d", L ->
item);
                                printf(" ,
");
                        L = L -> prox;
        } else {
                        printf("%d", L ->
item);
                        L = L -> prox;
                }
}
        printf("]");
}

int main(void) {
        printf(" **********
EXERCÍCIO 4. SOMA DOS ITENS DA
LISTA **********\n\n",
setlocale(LC_ALL, ""));

        Lista I = no(3, no(1, no(5,
NULL)));
        exibe(I);

        printf("\n\n\tTamanho =
%d\n", tamanho (I));
        printf("\n\tSoma dos itens =
%d", soma (I));

        printf("\n\n");
        system ("pause");
        return 0;
}
```

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

int tamanho (Lista L) {
	int t = 0;
	while (L) {
		t++;
		L = L -> prox;
	}
	return t;
}

int soma (Lista L) {
	int somatoria = 0;
	while (L != NULL) {
		somatoria = somatoria
+ L -> item;
		L = L -> prox;
	}
	return somatoria;
}

void anexa (Lista *A, Lista B) {
	if ( !B ) return;
		while ( *A )
			A = &( *A ) ->
prox;
		*A = B;
}

void exibe (Lista L) {
	printf("\t[");
```

```
********** EXERCÍCIO 5. PROGRAMA PARA ANEXAÇÃO *

       H =     [4 , 2]
       I =     [3 , 1 , 5]

       Pressione enter


       H =     [4 , 2 , 3 , 1 , 5]
       I =     [3 , 1 , 5]

Pressione qualquer tecla para continuar. . .
```

```
        while (L != NULL) {
                if (L -> prox != NULL){
                        printf("%d", L ->
item);
                                printf(" ,
");
                        L = L -> prox;
        } else {
                        printf("%d", L ->
item);
                        L = L -> prox;
                }
}
        printf("]");
}

int main(void) {
        printf(" **********
EXERCÍCIO 5. PROGRAMA PARA
ANEXAÇÃO **********\n\n",
setlocale(LC_ALL, ""));

        Lista H = no(4, no (2, NULL));
        Lista I = no(3, no(1, no(5,
NULL)));
        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);
        printf("\n\n\tPressione
enter\n\n"); getchar();
        anexa(&H,I);
        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);



        printf("\n\n");
        system ("pause");
        return 0;
}
```

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

void anexa (Lista *A, Lista B) {
	if ( !B ) return;
		while ( *A )
			A = &( *A ) ->
prox;
		*A = B;
}

void destroi (Lista *L) {
	while ( *L ) {
		Lista n = *L;
		*L = n -> prox;
		free(n);
	}
}

void exibe (Lista L) {
	printf("\t[");
	while (L != NULL) {
		if (L -> prox != NULL){
			printf("%d", L ->
item);
				printf(" ,
");
			L = L -> prox;
	} else {
			printf("%d", L ->
item);
			L = L -> prox;
```

```
********** EXERCÍCIO 6. PROGRAMA PARA DESTRUIÇÃO D

       H =     [4 , 2]
       I =     [3 , 1 , 5]

       Pressione 'enter' para anexar

       H =     [4 , 2 , 3 , 1 , 5]
       I =     [3 , 1 , 5]

       Pressione 'enter' para destruir

       H após a destruição =   []
       I após a destruição =   []

Pressione qualquer tecla para continuar. . .
```

```
                    }
}
        printf("]");
}

int main(void) {
        printf(" **********
EXERCÍCIO 6. PROGRAMA PARA
DESTRUIÇÃO DE LISTA
**********\n\n", setlocale(LC_ALL,
""));

        Lista H = no(4, no (2, NULL));
        Lista I = no(3, no(1, no(5,
NULL)));
        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);
        printf("\n\n\tPressione 'enter'
para anexar\n"); getchar();

        anexa(&H,I);

        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);
        printf("\n\n\tPressione 'enter'
para destruir"); getchar();

        destroi(&I);
        destroi(&H);

        printf("\n\tH após a destruição
= "); exibe(H);
        printf("\n\tI após a destruição
= "); exibe(I);

        printf("\n\n");
        system ("pause");
        return 0;
}
```

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

void anexa (Lista *A, Lista B) {
	if ( !B ) return;
		while ( *A )
			A = &( *A ) ->
prox;
		*A = B;
}

void destroi (Lista *L) {
	while ( *L ) {
		Lista n = *L;
		*L = n -> prox;
		free(n);
	}
}

void ultimo(Lista L){
		while( L != NULL ) {
		if(L->prox == NULL){
			printf("%d",L-
>item);
		}
		L = L->prox;
	}
}

void exibe (Lista L) {
	printf("\t[");
	while (L != NULL) {
```

```
********** EXERCÍCIO 7. ÚLTIMO ITEM DA LISTA **********

       H =     [4 , 2]
       I =     [3 , 1 , 5]

       Pressione 'enter' para anexar

       H =     [4 , 2 , 3 , 1 , 5]
       I =     [3 , 1 , 5]

       O último item da lista I é: 5
       O último item da lista H é: 5

       Pressione 'enter' para destruir

       H após a destruição =   []
       I após a destruição =   []

Pressione qualquer tecla para continuar. . .
```

```
                if (L -> prox != NULL){
                        printf("%d", L ->
item);
                                printf(" ,
");
                        L = L -> prox;
        } else {
                        printf("%d", L ->
item);
                        L = L -> prox;
                }
}
        printf("]");
}

int main(void) {
        printf(" **********
EXERCÍCIO 7. ÚLTIMO ITEM DA LISTA
**********\n\n", setlocale(LC_ALL,
""));

        Lista H = no(4, no (2, NULL));
        Lista I = no(3, no(1, no(5,
NULL)));
        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);
        printf("\n\n\tPressione 'enter'
para anexar\n"); getchar();

        anexa(&H,I);

        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);

        printf("\n\n\tO último item da
lista I é: ");
        ultimo(I);
        printf("\n\tO último item da
lista H é: ");
        ultimo(H);

        printf("\n\n\tPressione 'enter'
para destruir"); getchar();
        destroi(&I);
        destroi(&H);
```

```
        printf("\n\tH após a destruição
= "); exibe(H);
        printf("\n\tI após a destruição
= "); exibe(I);

        printf("\n\n");
        system ("pause");
        return 0; }
```

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

typedef int Item;
typedef struct no {
        Item item;
        struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
        Lista n = malloc(sizeof(struct
no));
        n -> item = x;
        n -> prox = p;
        return n;
}

void anexa (Lista *A, Lista B) {
        if ( !B ) return;
                while ( *A )
                        A = &( *A ) ->
prox;
                *A = B;
}

void destroi (Lista *L) {
        while ( *L ) {
                Lista n = *L;
                *L = n -> prox;
                free(n);
        }
}

void ultimo(Lista L){
                while( L != NULL ) {
                if(L->prox == NULL){
                        printf("%d",L-
>item);
```

```
********** EXERCÍCIO 8. PERTINÊNCIA EM LISTA *********

       H =     [4 , 2]
       I =     [3 , 1 , 5]

       Pressione 'enter' para anexar

       H =     [4 , 2 , 3 , 1 , 5]
       I =     [3 , 1 , 5]

       O último item da lista I é: 5
       O último item da lista H é: 5

       LISTA I
       [3 , 1 , 5]
       O número 7 não pertence à lista
       O número 1 pertence à lista
       O número 2 não pertence à lista

       LISTA H
       [4 , 2 , 3 , 1 , 5]
       O número 7 não pertence à lista
       O número 4 pertence à lista
       O número 2 pertence à lista

       Pressione 'enter' para destruir

       H após a destruição =   []
       I após a destruição =   []

Pressione qualquer tecla para continuar. . .
```

```
                }
                L = L->prox;
        }
}

int pertence (int x, Lista L){

        while( L != NULL ) {
                if( x == L->item  ){
                        printf("\n\tO
número %d pertence à lista", L-
>item);
                        return 0;
                }
                L = L->prox;
        }
        printf("\n\tO número %d não
pertence à lista", x);
}


void exibe (Lista L) {
        printf("\t[");
        while (L != NULL) {
                if (L -> prox != NULL){
                        printf("%d", L ->
item);
                                printf(" ,
");
                        L = L -> prox;
        } else {
                        printf("%d", L ->
item);
                        L = L -> prox;
                }
}
        printf("]");
}

int main(void) {
        printf(" **********
EXERCÍCIO 8. PERTINÊNCIA EM LISTA
**********\n\n", setlocale(LC_ALL,
""));

        Lista H = no(4, no (2, NULL));
```

```
        Lista I = no(3, no(1, no(5,
NULL)));
        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);
        printf("\n\n\tPressione 'enter'
para anexar\n"); getchar();

        anexa(&H,I);

        printf("\tH = "); exibe(H);
        printf("\n\tI = "); exibe(I);

        printf("\n\n\tO último item da
lista I é: ");
        ultimo(I);
        printf("\n\tO último item da
lista H é: ");
        ultimo(H);

        printf("\n\n\tLISTA I\n");

        exibe(I);
        pertence(7,I);
        pertence(1,I);
        pertence(2,I);


        printf("\n\n\tLISTA H\n");

        exibe(H);
        pertence(7,H);
        pertence(4,H);
        pertence(2,H);

        printf("\n\n\tPressione 'enter'
para destruir"); getchar();
        destroi(&I);
        destroi(&H);

        printf("\n\tH após a destruição
= "); exibe(H);
        printf("\n\tI após a destruição
= "); exibe(I);

        printf("\n\n");
        system ("pause");
        return 0; }
```

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

typedef int Item;
typedef struct no {
	Item item;
	struct no *prox;
} *Lista;

Lista no (Item x, Lista p) {
	Lista n = malloc(sizeof(struct
no));
	n -> item = x;
	n -> prox = p;
	return n;
}

void inversa(Lista L){

	if( L == NULL ) {
		return;
	}
	inversa(L->prox);
  printf("%d ",L->item);
}

void exibe (Lista L) {
	printf("\t[");
	while (L != NULL) {
		if (L -> prox != NULL){
			printf("%d", L ->
item);
				printf(" ,
");
			L = L -> prox;
	} else {
			printf("%d", L ->
item);
			L = L -> prox;
		}
}
	printf("]");
}

int main(void) {
```

```
********** EXERCÍCIO 9. INVERSÃO DE LIS

       [3 , 1 , 5]

       O inverso da lista é: [5 1 3 ]


Pressione qualquer tecla para continuar.
```

```
        printf(" **********
EXERCÍCIO 9. INVERSÃO DE LISTA
**********\n\n", setlocale(LC_ALL,
""));

        Lista I =
no(3,no(1,no(5,NULL)));
        exibe(I);
        printf("\n\n\tO inverso da lista
é: ");
        printf("[");
        inversa(I);
        printf("]\n");

        printf("\n\n");
        system ("pause");
        return 0;
}
```