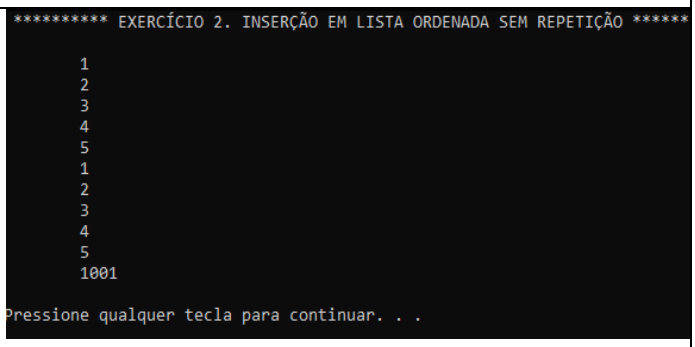
Nome: Rodrigo Silva de Oliveira **RA:** 1840482113026 **ADS – NOITE**

```
#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 ins (Item x, Lista *L) {
	while (*L != NULL &&
(*L) -> item < x)
		L = &(*L) ->
prox;
	*L = no (x,*L);
}

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

int main(void) {
	printf(" **********
EXERCÍCIO 1. PROGRAMA
PARA INSERÇÃO EM LISTA
ORDENADA **********\n\n",
setlocale(LC_ALL, ""));

	Lista l = NULL;
	ins(4,&l);
	ins(1,&l);
```

```
********** EXERCÍCIO 1. PROGRAMA PARA INSERÇÃO EM LISTA ORDENADA **********

	1
	2
	3
	4
	5

Pressione qualquer tecla para continuar. . .
```

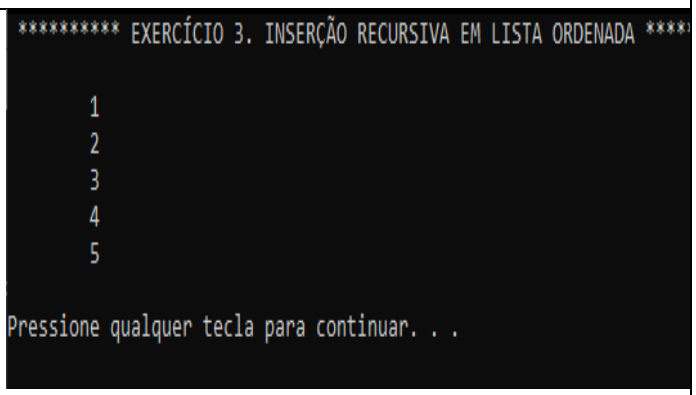
```
        ins(3,&I);
        ins(5,&I);
        ins(2,&I);
        exibe(I);

        printf("\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 ins (Item x, Lista *L) {
        while (*L != NULL &&
(*L) -> item < x)
                L = &(*L) ->
prox;
        *L = no (x,*L);
}

void ins_sr(Item x, Lista *L){
    Lista I = *L;
    while( *L != NULL ){
       if((*L)->item==x)
         return;
       L = &(*L)->prox;
    }
    ins(x, &I);
}

void exibe (Lista L) {
        while (L != NULL) {
```



```
                printf("\t%d\n",
L -> item);
                L = L -> prox;
        }
}

int main(void) {
        printf(" **********
EXERCÍCIO 2. INSERÇÃO EM
LISTA ORDENADA SEM
REPETIÇÃO **********\n\n",
setlocale(LC_ALL, ""));

        Lista I = NULL;
        ins(4,&I);
        ins(1,&I);
        ins(3,&I);
        ins(5,&I);
        ins(2,&I);
        exibe(I);
        ins_sr(1,&I);
        ins_sr(1001,&I);
        exibe(I);

        printf("\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 ins (Item x, Lista *L) {
```



```
        while (*L != NULL &&
(*L) -> item < x)
                L = &(*L) ->
prox;
        *L = no (x,*L);
}

void ins_rec(Item x, Lista *L) {
    if (*L != NULL && (*L)->item
< x){
      ins_rec(x, &(*L)->prox);
    }else{
      *L = no(x,*L);
    }
}

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

int main(void) {
        printf(" **********
EXERCÍCIO 3. INSERÇÃO
RECURSIVA EM LISTA
ORDENADA **********\n\n",
setlocale(LC_ALL, ""));

        Lista I = NULL;
        ins_rec(4,&I);
        ins_rec(1,&I);
        ins_rec(3,&I);
        ins_rec(5,&I);
        ins_rec(2,&I);
        exibe(I);

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

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

typedef int Item;
typedef struct no {
```

```
********** EXERCÍCIO 4. EXIBIÇÃO RECURSIVA CRESCENTE DE LISTA ORDENADA *****

        1       2       3       4       5

Pressione qualquer tecla para continuar. . .
```

```
        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 ins (Item x, Lista *L) {
        while (*L != NULL &&
(*L) -> item < x)
                L = &(*L) ->
prox;
        *L = no (x,*L);
}

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

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

int main(void) {
        printf(" **********
EXERCÍCIO 4. EXIBIÇÃO
RECURSIVA CRESCENTE DE
LISTA ORDENADA
**********\n\n",
setlocale(LC_ALL, ""));

        Lista I = NULL;
        ins(4,&I);
        ins(1,&I);
        ins(3,&I);
```

```
        ins(5,&I);
        ins(2,&I);
        exibe_crescente(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 ins (Item x, Lista *L) {
        while (*L != NULL &&
(*L) -> item < x)
                L = &(*L) ->
prox;
        *L = no (x,*L);
}

void rem(Item x, Lista *L) {
   while( *L != NULL && (*L)-
>item < x )
      L = &(*L)->prox;
   if( *L == NULL || (*L)->item
> x ) return;
   Lista n = *L;
   *L = n->prox;
   free(n);
}

void exibe_decrescente(Lista
L){
```

```
********** EXERCÍCIO 5. EXIBIÇÃO RECURSIVA DEC

       5       4       3       2       1

Pressione qualquer tecla para continuar. . .
```

```
    Lista I = L;
    if(I == NULL) return;
    while( L ->prox != NULL )
        L = L->prox;
    printf("\t%d",L->item);
    rem(L->item, &I);
    exibe_decrescente(I);
}

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

int main(void) {
        printf(" **********
EXERCÍCIO 5. EXIBIÇÃO
RECURSIVA DECRESCENTE DE
LISTA ORDENADA
**********\n\n",
setlocale(LC_ALL, ""));

        Lista I = NULL;
        ins(4,&I);
        ins(1,&I);
        ins(3,&I);
        ins(5,&I);
        ins(2,&I);
        exibe_decrescente(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;
```

```
********** EXERCÍCIO 6. PROGRAMA PARA REMOÇÃO EM LISTA OR

        1
        2
        4
        5


Pressione qualquer tecla para continuar. . .
```

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

void ins (Item x, Lista *L) {
	while (*L != NULL &&
(*L) -> item < x)
		L = &(*L) ->
prox;
	*L = no (x,*L);
}

void rem(Item x, Lista *L) {
   while( *L != NULL && (*L)-
>item < x )
      L = &(*L)->prox;
   if( *L == NULL || (*L)->item
> x ) return;
   Lista n = *L;
   *L = n->prox;
   free(n);
}

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

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

	Lista I = NULL;
	ins(4,&I);
	ins(1,&I);
	ins(3,&I);
	ins(5,&I);
```

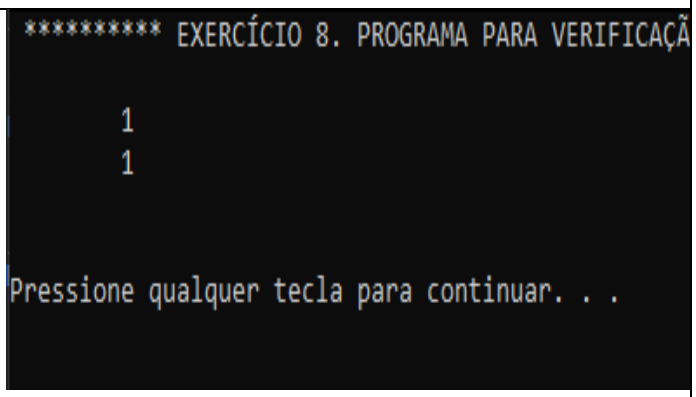
```
        ins(2,&l);
        rem(3, &l);
        exibe(l);

        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 ins (Item x, Lista *L) {
        while (*L != NULL &&
(*L) -> item < x)
                L = &(*L) ->
prox;
        *L = no (x,*L);
}

void rem_todo (Item x, Lista*
L) {
  while( *L != NULL && (*L)-
>item < x )
    L = &(*L)->prox;
  while(*L != NULL){
    if( *L == NULL || (*L)->item
> x ) return;
    Lista n = *L;
    *L = n->prox;
    free(n);
  }
}
```

```
********** EXERCÍCIO 7. REMOÇÃO DE TODAS AS OCORRÊNCIAS EM LI

Antes da remoção do item '3':
        1
        2
        3
        4
        5

Depois da remoção do item '3':
        1
        2
        4
        5


Pressione qualquer tecla para continuar. . .
```

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

int main(void) {
	printf(" **********
EXERCÍCIO 7. REMOÇÃO DE
TODAS AS OCORRÊNCIAS EM
LISTA ORDENADA
**********\n\n",
setlocale(LC_ALL, ""));

	Lista I = NULL;
	ins(4,&I);
	ins(1,&I);
	ins(3,&I);
	ins(5,&I);
	ins(2,&I);
	printf("Antes da
remoção do item '3':\n");
   exibe(I);
   rem_todo(3,&I);
   printf("\nDepois da remoção
do item '3':\n");
   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 ins (Item x, Lista *L) {
	while (*L != NULL &&
(*L) -> item < x)
		L = &(*L) ->
prox;
	*L = no (x,*L);
}

int pert (Item x, Lista L) {
	while (L != NULL && L -
> item < x)
		L = L -> prox;
	return (L != NULL && L -
> item == x);
}

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

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

	Lista I = NULL;
	ins(4,&I);
	ins(1,&I);
	ins(3,&I);
	ins(5,&I);
	ins(2,&I);
	printf("\t%d\n",
pert(5,I));
```

```
        printf("\t%d\n",
pert(3,l));

        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 ins (Item x, Lista *L) {
        while (*L != NULL &&
(*L) -> item < x)
                L = &(*L) ->
prox;
        *L = no (x,*L);
}

int pert_rec(Item x,Lista L){
   if (L != NULL && L->item < x )
      return pert_rec(x,L-
>prox);
   return (L != NULL && L-
>item == x);
}

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

```
********** EXERCÍCIO 9. VERIFICAÇÃO DE PERTINÊ

        1
        1


Pressione qualquer tecla para continuar. . .
```

```
}

int main(void) {
	printf(" **********
EXERCÍCIO 9. VERIFICAÇÃO DE
PERTINÊNCIA RECURSIVA
**********\n\n",
setlocale(LC_ALL, ""));

	Lista I = NULL;
	ins(4,&I);
	ins(1,&I);
	ins(3,&I);
	ins(5,&I);
	ins(2,&I);
	printf("\t%d\n",
pert_rec(3,I));
	printf("\t%d\n",
pert_rec(5,I));

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