

Exercício 1

The screenshot shows a C program in a code editor and its execution output in a terminal window.

Code Editor (Ex1.c):

```
4 typedef int Item;
5 typedef struct no {
6     Item item;
7     struct no *prox;
8 } *Lista;
9
10 void exibe(Lista L) {
11     while( L != NULL ) {
12         printf("%d\n", L->item);
13         L = L->prox;
14     }
15 }
16
17 Lista no(Item x, Lista p) {
18     Lista n = malloc(sizeof(struct no));
19     n->item = x;
20     n->prox = p;
21     return n;
22 }
23
24
25 void ins(Item x, Lista *L) {
26     while( *L != NULL && (*L)->item < x )
27         L = &(*L)->prox;
28     *L = no(x, *L);
29 }
30
31 int main(void) {
32     Lista I = NULL;
33     ins(4, &I);
34     ins(1, &I);
35     ins(3, &I);
36     ins(5, &I);
37     ins(2, &I);
38     exibe(I);
39     return 0;
40 }
```

Terminal Window:

```
C:\Users\P08012631\Desktop\ >
1
2
3
4
5
-----
Process exited after 0.6592 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 2

The screenshot shows a C program in a code editor and its execution output in a terminal window.

Code Editor (Ex2.c):

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef int Item;
5 typedef struct no {
6     Item item;
7     struct no *prox;
8 } *Lista;
9
10 void exibe(Lista L) {
11     while( L != NULL ) {
12         printf("%d\n", L->item);
13         L = L->prox;
14     }
15 }
16
17 Lista no(Item x, Lista p) {
18     Lista n = malloc(sizeof(struct no));
19     n->item = x;
20     n->prox = p;
21     return n;
22 }
23
24 void ins(Item x, Lista *L) {
25     while( *L != NULL && (*L)->item < x )
26         L = &(*L)->prox;
27     *L = no(x, *L);
28 }
29
30 void ins_isr(Item x, Lista *L) {
31     while( *L != NULL && (*L)->item < x )
32         L = &(*L)->prox;
33 }
34
35 if( *L != NULL && (*L)->item == x ) {
```

Terminal Window:

```
C:\Users\cloud\OneDrive - Fat >
1
2
3
4
5
6
-----
Process exited after 0.09834 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 3

```
Ex2.c  Ex4.c  Ex3.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef int Item;
5  typedef struct no {
6      Item item;
7      struct no *prox;
8  } *Lista;
9
10 void exibe(Lista L) {
11     while (L != NULL) {
12         printf("%d\n", L->item);
13         L = L->prox;
14     }
15 }
16
17 Lista no(Item x, Lista p) {
18     Lista n = malloc(sizeof(struct no));
19     n->item = x;
20     n->prox = p;
21     return n;
22 }
23
24 void ins_rec(Item x, Lista *L) {
25     if (*L == NULL || (*L->item > x) {
26         *L = no(x, *L);
27     } else {
28         ins_rec(x, &(*L->prox));
29     }
30 }
31
32 int main(void) {
33     int num=0;
34     Lista I = NULL;
35     ins_rec(4, &I);
36     ins_rec(1, &I);
```

```
C:\Users\cloud\OneDrive - Fat  X  +  v
1
2
3
4
5
10

Informe um numero para inserir na lista:
8

Lista ordenada:
1
2
3
4
5
8
10

-----
Process exited after 5.492 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 4

```
Ex2.c  Ex3.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef int Item;
5  typedef struct no {
6      Item item;
7      struct no *prox;
8  } *Lista;
9
10 void exibe(Lista L) {
11     while (L != NULL) {
12         printf("%d\n", L->item);
13         L = L->prox;
14     }
15 }
16
17 Lista no(Item x, Lista p) {
18     Lista n = malloc(sizeof(struct no));
19     n->item = x;
20     n->prox = p;
21     return n;
22 }
23
24 void ins(Item x, Lista *L) {
25     while (*L != NULL && (*L->item < x)
26         L = &(*L->prox);
27     *L = no(x, *L);
28 }
29
30 void ins_rsr(Item x, Lista *L) {
31     if (*L == NULL || (*L->item > x) {
32         *L = no(x, *L);
33     } else if ((*L->item < x) {
34         ins_rsr(x, &(*L->prox));
35     }
```

```
C:\Users\cloud\OneDrive - Fat  X  +  v
1
2
3
4
5
6

-----
Process exited after 0.08435 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 5

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef int Item;
5 typedef struct no {
6     Item item;
7     struct no *prox;
8 } *Lista;
9
10 Lista no(Item x, Lista p) {
11     Lista n = malloc(sizeof(struct no));
12     n->item = x;
13     n->prox = p;
14     return n;
15 }
16
17 void ins_rec(Item x, Lista *L) {
18     if (*L == NULL || (*L)->item > x) {
19         *L = no(x, *L);
20     } else {
21         ins_rec(x, &(*L)->prox);
22     }
23 }
24
25 void exibe(Lista L) {
26     while (L != NULL) {
27         printf("%d\n", L->item);
28         L = L->prox;
29     }
30 }
31
32 int main(void) {
33     Lista I = NULL;
34     ins_rec(4, &I);
35     ins_rec(1, &I);
36     ins_rec(3, &I);
```

```
C:\Users\cloud\OneDrive - Fat  X + v
1
2
3
4
5
-----
Process exited after 0.07709 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 6

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef int Item;
5 typedef struct no {
6     Item item;
7     struct no *prox;
8 } *Lista;
9
10 Lista no(Item x, Lista p) {
11     Lista n = malloc(sizeof(struct no));
12     n->item = x;
13     n->prox = p;
14     return n;
15 }
16
17 void ins_rec(Item x, Lista *L) {
18     if (*L == NULL || (*L)->item < x) {
19         *L = no(x, *L);
20     } else {
21         ins_rec(x, &(*L)->prox);
22     }
23 }
24
25 void exibe(Lista L) {
26     while (L != NULL) {
27         printf("%d\n", L->item);
28         L = L->prox;
29     }
30 }
31
32 int main(void) {
33     Lista I = NULL;
34     ins_rec(4, &I);
35     ins_rec(1, &I);
36     ins_rec(3, &I);
```

```
C:\Users\cloud\OneDrive - Fat  X + v
5
4
3
2
1
-----
Process exited after 0.1027 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 7

```
Ex1.c Ex3.c Ex2.c Ex4.c Ex5.c Ex6.c Ex7.c
7 | struct no *prox;
8 | } *Lista;
9 |
10 | void exibe(Lista L) {
11 |     while( L != NULL ) {
12 |         printf("%d\n",L->item);
13 |         L = L->prox;
14 |     }
15 | }
16 |
17 | Lista no(Item x, Lista p) {
18 |     Lista n = malloc(sizeof(struct no));
19 |     n->item = x;
20 |     n->prox = p;
21 |     return n;
22 | }
23 |
24 | void ins(Item x, Lista *L) {
25 |     while( *L != NULL && (*L)->item < x )
26 |         L = &(*L)->prox;
27 |     *L = no(x,*L);
28 | }
29 |
30 | void rem(Item x, Lista *L) {
31 |     while( *L != NULL && (*L)->item < x )
32 |         L = &(*L)->prox;
33 |     if( *L == NULL || (*L)->item > x ) return;
34 |     Lista n = *L;
35 |     *L = n->prox;
36 |     free(n);
37 | }
38 |
39 | int main(void) {
40 |     Lista l = NULL;
41 |     ins(4.&T); ins(1.&T); ins(3.&T); ins(5.&T); ins(2.&T);
```

```
C:\Users\cloud\OneDrive - Fat x + v
1
2
4
5

-----
Process exited after 0.1165 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 8

```
Ex1.c Ex3.c Ex2.c Ex4.c Ex5.c Ex6.c Ex7.c
1 | #include <stdio.h>
2 | #include <stdlib.h>
3 |
4 | typedef int Item;
5 | typedef struct no {
6 |     Item item;
7 |     struct no *prox;
8 | } *Lista;
9 |
10 | void exibe(Lista L) {
11 |     while( L != NULL ) {
12 |         printf("%d\n",L->item);
13 |         L = L->prox;
14 |     }
15 | }
16 |
17 | Lista no(Item x, Lista p) {
18 |     Lista n = malloc(sizeof(struct no));
19 |     n->item = x;
20 |     n->prox = p;
21 |     return n;
22 | }
23 |
24 | void ins(Item x, Lista *L) {
25 |     while( *L != NULL && (*L)->item < x )
26 |         L = &(*L)->prox;
27 |     *L = no(x,*L);
28 | }
29 |
30 | void rem(Item x, Lista *L) {
31 |     while( *L != NULL && (*L)->item < x )
32 |         L = &(*L)->prox;
33 |     if( *L == NULL || (*L)->item > x ) return;
34 |     Lista n = *L;
35 |     *L = n->prox;
36 |     free(n);
37 | }
```

```
C:\Users\cloud\OneDrive - Fat x + v
Lista antes de remover:
1
2
3
3
3
4
5

Lista depois de remover todas as ocorrencias de 3:
1
2
4
5

-----
Process exited after 0.1074 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 9

```
bx1.c  bx3.c  bx2.c  bx4.c  bx5.c  bx6.c  bx7.c  bx8.c  bx9.c
36 |     free(n);
37 | }
38 |
39 | void rem_todo(Item x, Lista *L) {
40 |     while (*L != NULL) {
41 |         if ((*L)->item < x) {
42 |             L = &(*L)->prox;
43 |         } else if ((*L)->item == x) {
44 |             Lista n = *L;
45 |             *L = n->prox;
46 |             free(n);
47 |         } else {
48 |             break;
49 |         }
50 |     }
51 | }
52 |
53 | int pert(Item x, Lista L) {
54 |     while( L != NULL && L->item < x )
55 |         L = L->prox;
56 |     return (L != NULL && L->item == x);
57 | }
58 |
59 |
60 | int main(void) {
61 |     Lista I = NULL;
62 |     ins(4,&I);
63 |     ins(1,&I);
64 |     ins(3,&I);
65 |     ins(2,&I);
66 |     printf("%d\n", pert(5,I));
67 |     printf("%d\n", pert(3,I));
68 |     return 0;
69 | }
70 |
```

```
C:\Users\cloud\OneDrive - Fat X + v
0
1
-----
Process exited after 0.1045 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 10

```
bx1.c  bx3.c  bx2.c  bx4.c  bx5.c  bx6.c  bx7.c  bx8.c  bx9.c  bx10.c
45 |         *L = n->prox;
46 |         free(n);
47 |     } else {
48 |         break;
49 |     }
50 | }
51 |
52 |
53 | int pert_rec(Item x, Lista L) {
54 |     if (L == NULL) {
55 |         return 0;
56 |     } else if (L->item == x) {
57 |         return 1;
58 |     } else if (L->item > x) {
59 |         return 0;
60 |     } else {
61 |         return pert_rec(x, L->prox);
62 |     }
63 | }
64 |
65 | int main(void) {
66 |     Lista I = NULL;
67 |     ins(4, &I);
68 |     ins(1, &I);
69 |     ins(3, &I);
70 |     ins(2, &I);
71 |
72 |
73 |     printf("Usando pertinencia recursiva (0=Falso, 1=Verdadeiro):\n");
74 |     printf("5 esta na lista? %d\n", pert_rec(5, I));
75 |     printf("3 esta na lista? %d\n", pert_rec(3, I));
76 |
77 |     return 0;
78 | }
79 |
```

```
C:\Users\cloud\OneDrive - Fat X + v
Usando pertinencia recursiva (0=Falso, 1=Verdadeiro):
5 esta na lista? 0
3 esta na lista? 1
-----
Process exited after 0.1258 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 11

```
Ex1.c  Ex3.c  Ex2.c  Ex4.c  Ex5.c  Ex6.c  Ex7.c  Ex8.c  Ex9.c  Ex10.c  Ex11.c
27 | *L = no(x, *L);
28 | }
29 |
30 | int osc(Lista L) {
31 |     if (L == NULL || L->prox == NULL) {
32 |         return 1;
33 |     }
34 |     if (L->item > L->prox->item) {
35 |         return 0;
36 |     }
37 |     return osc(L->prox);
38 | }
39 |
40 | int main(void) {
41 |     Lista I = NULL;
42 |     ins(1, &I);
43 |     ins(2, &I);
44 |     ins(3, &I);
45 |     ins(4, &I);
46 |
47 |     printf("Lista antes de verificar se está ordenada:");
48 |     exibe(I);
49 |
50 |     printf("A lista está ordenada de forma crescente? ");
51 |
52 |     ins(3, &I);
53 |
54 |     printf("\nLista depois de inserir um valor fora de ordem:");
55 |     exibe(I);
56 |
57 |     printf("A lista está ordenada de forma crescente? ");
58 |
59 |     return 0;
60 | }
61 |
```

```
C:\Users\cloud\OneDrive - Fat X + v
Lista antes de verificar se está ordenada:
1
2
3
4
A lista está ordenada de forma crescente? 1

Lista depois de inserir um valor fora de ordem:
1
2
3
3
4
A lista está ordenada de forma crescente? 1

-----
Process exited after 0.1266 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 12

```
Ex1.c  Ex3.c  Ex4.c  Ex5.c  Ex6.c  Ex7.c  Ex8.c  Ex9.c  Ex10.c  Ex11.c  Ex12.c
22 | }
23 |
24 | void ins(Item x, Lista *L) {
25 |     while (*L != NULL && (*L)->item < x)
26 |         L = &(*L)->prox;
27 |     *L = no(x, *L);
28 | }
29 |
30 | int oec(Lista L) {
31 |     if (L == NULL || L->prox == NULL) {
32 |         return 1;
33 |     }
34 |     if (L->item >= L->prox->item) {
35 |         return 0;
36 |     }
37 |     return oec(L->prox);
38 | }
39 |
40 | int main(void) {
41 |     Lista I = NULL;
42 |     ins(1, &I);
43 |     ins(2, &I);
44 |     ins(3, &I);
45 |     ins(4, &I);
46 |
47 |     printf("Lista antes de verificar se está ordenada:\n");
48 |     exibe(I);
49 |
50 |     printf("A lista está ordenada de forma crescente? %d\n", oec(I));
51 |
52 |     ins(3, &I);
53 |
54 |     printf("\nLista depois de inserir um valor fora de ordem:\n");
55 |     exibe(I);
56 |
```

```
C:\Users\cloud\OneDrive - Fat X + v
Lista antes de verificar se está ordenada:
1
2
3
4
A lista está ordenada de forma crescente? 1

Lista depois de inserir um valor fora de ordem:
1
2
3
3
4
A lista está ordenada de forma crescente? 0

-----
Process exited after 0.1083 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exercício 13

```
Ex1.c Ex3.c Ex2.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c
6   Item item;
7   struct no *prox;
8 } *Lista;
9
10 void exibe(Lista L) {
11     while (L != NULL) {
12         printf("%d\n", L->item);
13         L = L->prox;
14     }
15 }
16
17 Lista no(Item x, Lista p) {
18     Lista n = malloc(sizeof(struct no));
19     n->item = x;
20     n->prox = p;
21     return n;
22 }
23
24 void ins(Item x, Lista *L) {
25     while (*L != NULL && (*L->item > x))
26         L = &(*L->prox);
27     *L = no(x, *L);
28 }
29
30 int osd(Lista L) {
31     if (L == NULL || L->prox == NULL) {
32         return 1;
33     }
34     if (L->item < L->prox->item) {
35         return 0;
36     }
37     return osd(L->prox);
38 }
39
40 int main(void) {
41     Lista I = NULL;
```

```
C:\Users\cloud\OneDrive - Fat X + v
Lista antes de verificar se esta ordenada:
4
3
2
1
A lista esta ordenada de forma decrescente? 1

Lista depois de inserir um valor fora de ordem:
4
3
3
2
1
A lista esta ordenada de forma decrescente? 1

-----
Process exited after 0.1265 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

Exercício 14

```
Ex1.c Ex3.c Ex2.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c
27 *L = no(x, *L);
28 }
29
30 int osd(Lista L) {
31     if (L == NULL || L->prox == NULL) {
32         return 1;
33     }
34     if (L->item <= L->prox->item) {
35         return 0;
36     }
37     return osd(L->prox);
38 }
39
40 int main(void) {
41     Lista I = NULL;
42     ins(4, &I);
43     ins(3, &I);
44     ins(2, &I);
45     ins(1, &I);
46
47     printf("Lista antes de verificar se esta ordenada\n");
48     exibe(I);
49
50     printf("A lista esta ordenada de forma decrescente? ");
51
52     ins(3, &I);
53
54     printf("\nLista depois de inserir um valor fora de ordem\n");
55     exibe(I);
56
57     printf("A lista esta ordenada de forma decrescente? ");
58
59     return 0;
60 }
61
```

```
C:\Users\cloud\OneDrive - Fat X + v
Lista antes de verificar se esta ordenada:
4
3
2
1
A lista esta ordenada de forma decrescente? 1

Lista depois de inserir um valor fora de ordem:
4
3
3
2
1
A lista esta ordenada de forma decrescente? 0

-----
Process exited after 0.1685 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```