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

struct listaNo {
    int data; //cada listaNo cantém um número
    struct listaNo *proximoPonteiro; //ponteiro para o próximo nó
};

typedef struct listaNo ListaNo;
typedef ListaNo *ListaNoPonteiro;

void inserir (ListaNoPonteiro *sPtr, int valor);
int excluir (ListaNoPonteiro *sPtr, int valor);
int estaVazia (ListaNoPonteiro sPtr);
void imprimirLista (ListaNoPonteiro atualPtr);
void instrucoes (void);

int main()
{
    ListaNoPonteiro inicioPtr = NULL; //inicialmente não existem nós
    int escolha; //escolha do usuário
    int dado; //valor inserido pelo usuário

    instrucoes(); //exibe o menu
    printf ("?\n");
    scanf("%d", &escolha);

    while (escolha != 4) {
        switch (escolha) {
        case 1: //inserir na lista
            printf ("Digite um numero: ");
            scanf ("%d", &dado);
            inserir(&inicioPtr, dado);
            break;
        case 2: //excluir da lista
            if (!estaVazia(inicioPtr)) { //se a lista não estiver vazia
                printf ("Digite o valor a ser excluido: ");
                scanf ("%d", &dado);
                //se o valor for encontrado, é removido
                if (excluir(&inicioPtr, dado)) {
                    printf ("%d excluido.\n", dado);
                }
                else {
                    printf ("O valor %d nao foi encontrado.\n", dado);
                }
            }
            else {
                printf ("A lista esta vazia.\n\n");
            }
            break;
        case 3:
            imprimirLista(inicioPtr);
            break;
        default:
            printf ("Escolha invalida.\n\n");
            instrucoes();
            break;
        }
        printf ("?\n");
        scanf ("%d", &escolha);
    }

    printf ("Fim da execucao!\n\n");
    return 0;
}

```

```c
//imprime as instruções na tela
void instrucoes(void) {
    printf ("Digite sua escolha:\n"
            "   1 para inserir um elemento na lista.\n"
            "   2 para excluir um elemento da lista.\n"
            "   3 para imprimir a lista.\n"
            "   4 para terminar");
}

//insere um novo elemento na lista
void inserir (ListaNoPonteiro *sPtr, int valor) {
    ListaNoPonteiro novoPtr; //ponteiro para o novo nó
    ListaNoPonteiro anteriorPtr; //ponteiro para o nó anterior na lista
    ListaNoPonteiro atualPtr; //ponteiro para o nó atual na lista

    novoPtr = malloc (sizeof (ListaNo)); //cria o nó

    if (novoPtr != NULL) { //tudo ok!
        novoPtr -> data = valor; //coloca o valor no nó
        novoPtr -> proximoPonteiro = NULL; //nó não se une a outro nó

        anteriorPtr = NULL;
        atualPtr = *sPtr;

        //loop para achar o local correto na lista
        while (atualPtr != NULL && valor > atualPtr -> data) {
            anteriorPtr = atualPtr; //caminha para...
            atualPtr = atualPtr -> proximoPonteiro; // ... próximo nó
        }

        //insere novo nó no início da lista
        if (anteriorPtr == NULL) {
            novoPtr -> proximoPonteiro = *sPtr;
            *sPtr = novoPtr;
        }
        else { //insere novo nó entre anteriorPtr e atualPtr
            anteriorPtr -> proximoPonteiro = novoPtr;
            novoPtr -> proximoPonteiro = atualPtr;
        }
    }
    else {
        printf ("%d nao inserido. Sem memoria disponivel.\n", valor);
    }
}

int excluir (ListaNoPonteiro *sPtr, int valor) {
    ListaNoPonteiro anteriorPtr; //ponteiro para o nó anterior na lista
    ListaNoPonteiro atualPtr; // ponteiro para o nó atual na lista
    ListaNoPonteiro tempPtr; //ponteiro para um nó temporário

    //exclui o primeiro nó
    if (valor == (*sPtr) -> data) {
        tempPtr = *sPtr; //aponta para o nó que está sendo removido
        *sPtr = (*sPtr) -> proximoPonteiro; //retira thread do nó
        free (tempPtr); //libera o nó com thread retirado
        return valor;
    }
    else {
        anteriorPtr = *sPtr;
        atualPtr = (*sPtr) -> proximoPonteiro;

        //loop para achar local correto na lista
        while (atualPtr != NULL && atualPtr -> data != valor) {
            anteriorPtr = atualPtr; //caminha até...
            atualPtr = atualPtr -> proximoPonteiro; // ...próximo nó
        }
```

```
133
134         //exclui nó em atualPtr;
135         if (atualPtr != NULL) {
136             tempPtr = atualPtr;
137             anteriorPtr -> proximoPonteiro = atualPtr -> proximoPonteiro;
138             free (tempPtr);
139             return valor;
140         }
141     }
142     return 0;
143 }
144
145 //retorna 1 se a lista estiver vazia, 0 se estiver cheia
146 int estaVazia (ListaNoPonteiro sPtr) {
147     return sPtr == NULL;
148 }
149
150 void imprimirLista (ListaNoPonteiro atualPtr) {
151     //se a lista estiver vazia
152     if (atualPtr == NULL) {
153         printf("Lista esta vazia.\n\n");
154     }
155     else {
156         printf("A lista e:\n");
157         //enquanto não chegar ao final da lista
158         while (atualPtr != NULL) {
159             printf("%d --> ", atualPtr -> data);
160             atualPtr = atualPtr -> proximoPonteiro;
161         }
162         printf("NULL\n\n");
163     }
164 }
165
```