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

struct el {             // Struktur fuer Listenelement
       int value;       // Wert des Elements
       struct el *next; // Zeiger auf nächstes Element
};

// Einfuegen von neuem Listenelement *p am Anfang nach start
void Insertfirst(struct el **start, struct el *p)
{
      p->next = *start;   // Nachfolger von p bisheriges 1.Element
      *start = p;         // start zeigt auf eingefuegtes *p
}

// Einfuegen von neuem Listenelement *p am Listenende
void Insertlast(struct el **start, struct el *p)
{
      struct el *t = 0;    // Hilfszeiger t auf Listenelement
      if(*start == 0){     // Liste leer ?
            *start = p;    // p nach start einfuegen
            p->next = 0;   // kein Nachfolger von p
            return;        // Verlassen der Funktion
      }
                           // Liste ist nicht leer
      t = *start;          // t zeigt auf 1. Listenelement
      while(t->next != 0)  // Suche t ohne Nachfolger
            t = t->next;   // t := Nachfolger von t
      t->next = p;         // letztes Element t mit Nachfolger p
      p->next = 0;         // p ohne Nachfolger
}

// Entfernen des ersten Listenelementes nach start
void Removefirst(struct el **start)
{
      if(*start != 0){                     // Existieren Listenelemente ?
            struct el *t = (*start)->next; // Hilfszeiger t zeigt auf 2.Element
            free(*start);                  // Freigabe 1.Element
            *start = t;                   // Start zeigt auf bisheriges 2.Element
      }
}

// Entfernen des letzten Elementes
void Removelast(struct el **start)
{
      struct el *v = *start, *t = 0; // v, t sind Hilfszeiger auf Listenelemente
      if(*start == 0) return;        // Wenn Liste leer, dann return
      if((*start)->next==0){         // Wenn nur 1 Element in Liste
            free(*start);           // Freigabe 1.Element
            *start = 0;             // start mit leerer Liste
            return;                 // Verlassen der Funktion
```

```
	}

	t = v->next;                // t zeigt auf 2.Element
	while(t->next != 0){        // Existiert Nachfolger von t ?
		v=t;                // Retten von t in v
		t=t->next;          // t zeigt auf Nachfolger von t
	}
	free(t);                    // letztes Element t freigeben
	v->next = 0;                // Vorgaenger von t ist neues letztes Element
}

// Berechnung Anzahl der Listenelemente
unsigned long Anzahl(struct el *start)
{
	struct el *t = start;     // t zeigt auf 1.Element
	unsigned long anz = 0UL;  // Anzahl mit 0 initialisieren
	if(start == 0)            // Wenn Liste leer, dann
		return anz;       // Rueckgabe anz = 0
	while(t->next != 0){      // Solange Nachfolger von t existiert
		anz++;            // inkrementiere anz um 1
		t=t->next;        // t zeigt auf seinen Nachfolger
	}
	return anz;               // Rueckgabe anz
}

// Anzeige aller Listenelemente
void Show(struct el *start)
{
	struct el *t = start;                   // t zeigt auf erstes Element
	printf("Ausgabe der Liste mit %d Elementen\n", Anzahl(start));
	while(t != 0){                          // Solange t auf Element zeigt
		printf("value = %d\n", t->value); // Ausgabe von value von t
		t = t->next;                    // t zeigt auf seinen Nachfolger
	}
	printf("End of show\n");
}

// Entferen aller Elemente aus der Liste
void Removeall(struct el **start)
{
	struct el *t = *start;          // t zeigt auf erstes Element
	while(*start != 0)              // solange Liste nicht leer ist
	{
		*start = (*start)->next;  // start zeigt auf 2.Element
		free(t);                  // Freigabe 1.Element
		t = *start;               // t zeigt auf neues 1.Element
	}
	printf("\nAll removed\n");
}
```

```
void main()                    // Haupteintrittspunkt fuer Abarbeitung
{
       struct el *start = 0; // initialisiere die Liste mit 0 (leere liste)

       // erzeuge neues Listenelement *p auf heap
       struct el *p = (struct el *)malloc(sizeof(struct el));
       p->value = 5;  // value = 5
       p->next = 0;   // kein Nachfolger

       Insertfirst(&start, p); // Einfuegen von p an den Anfang der Liste

       // erzeuge neues Listenelement *p auf heap
       p = (struct el *)malloc(sizeof(struct el));
       p->value = 3;   // value = 3
       p->next = 0;    // kein Nachfolger

       Insertfirst(&start, p); // Einfuegen von p an den Anfang der Liste

       // erzeuge neues Listenelement *p auf heap
       p = (struct el *)malloc(sizeof(struct el));
       p->value = 2;   // value = 2
       p->next = 0;    // kein Nachfolger

       Insertlast(&start, p);  // Einfuegen von p an das Ende der Liste

       Show(start);            // Anzeige Liste

       Removefirst(&start);    // Entferne 1.Element

       Removelast(&start);     // Entferen letztes Element

       printf("\nAnzahl Listenelemente anz = %d\n", Anzahl(start));

       Show(start);            // Anzeige Liste

       Removeall(&start);      // Leeren der Liste

       getchar();              // Warten auf Tasteneingabe
}
/*
Ausgabe der Liste mit 2 Elementen
value = 3
value = 5
value = 2
End of show
Anzahl Listenelemente anz = 0
Ausgabe der Liste mit 0 Elementen
value = 5
End of show
All removed
*/
```