Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TU Kaiserslautern
Fachbereich Informatik

Lösungshinweise zum Übungsblatt 13: Programmieren in C (WS 2018/19)

1. Die Aufgaben auf diesem Übungsblatt sind alle freiwillig zu bearbeiten. Falls Sie bei Blatt 11 die nötige Punktzahl nicht erreicht haben, können Sie die fehlenden Punkte hier nachholen. Bei Fragen wenden Sie sich bitte direkt an `pinc-support@cs.uni-kl.de`.
2. Die Abgabe erfolgt über das Exclaim-System gemeinsam mit Ihrem Teampartner.
3. Zur Beantwortung von Fragen und Hilfestellung bei der Bearbeitung kommen Sie bitte in die wöchentliche Fragestunde, Mittwoch 15:30, in Terminalraum 32-410.
4. Bitte laden Sie einzelne Dateien hoch, wie in der Aufgabenstellung angegeben, keine Archive, keine kompilierten Dateien.
5. Bitte beachten Sie unsere Regelung bei Plagiaten:
   - Wenn Sie sekundäre Quellen, wie Bücher oder das Internet verwenden, müssen Sie immer die Quelle angeben. Das einfache Kopieren aus anderen Quellen ist für die Übungen nicht gestattet. Wenn wir in einer Übungsabgabe kopierten Code ohne Quellenangabe finden, wird die gesamte Abgabe mit 0 Punkten bewertet.
   - Sie können Übungsaufgaben gerne mit den Mitgliedern anderer Teams diskutieren. Sie sollten jedoch Ihren Code vor der Abgabefrist nicht an andere Teams weitergeben bzw. zeigen.
   - Wenn Code von anderen Teams kopiert wurde, werden die Abgaben **von beiden Teams** mit 0 Punkten bewertet.
   - Wir behalten uns vor Punkte auch nachträglich abzuziehen, wenn ein Verstoß erst später bemerkt wird.

Aufgabe 1 *Programmverständnis (5 Punkte)*

Gegeben sind die folgenden C-Strukturen zur Darstellung von einfach verketteten Listen:

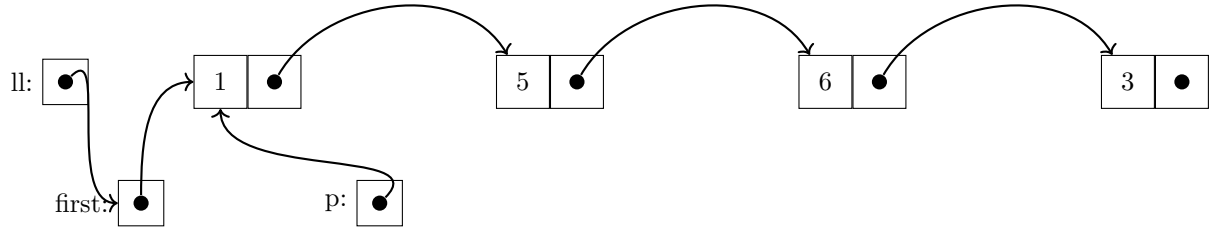
```c
typedef struct node node_t;
struct node {
  int value;
  node_t *next;
};
```

```c
typedef struct list list_t;
struct list {
  node_t *first;
};
```

Erklären Sie, was die folgende C-Funktion `f` macht. Verwenden Sie dazu die Eingabeliste mit den Elementen `1, 5, 6, 3` und zeichnen Sie den Zustand der Liste nach Ausführen der Funktion. Der Zustand vor der Ausführung der Schleife ist unten bereits vorgegeben.

```c
node_t * f(list_t *ll) {
  node_t *p;
  node_t *tmp;

  if (ll->first == NULL) {
      return;
  }
  p = ll->first;
  if (p->next == NULL) {
      ll->first = NULL;
  } else {
    while (p->next->next != NULL) {
        p = p->next;
    }
    tmp = p->next;
    p->next = p->next->next;
    p = tmp;
  }
  return p;
}
```



Lösung 1

Die Funktion entfernt den letzten Knoten aus der Liste und gibt einen Pointer auf diesen Knoten zurück.