Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für theoretische Informatik
Dr. B. Gärtner

25. Januar 2005

Informatik für Mathematiker und Physiker Serie 13 WS 04/05

URL: http://www.ti.inf.ethz.ch/ew/courses/Info1_04/

Aufgabe 1 [Schnellübung – 20 Min.] (5 Punkte)

Gruppe A–H

Implementieren Sie folgende Funktion:

```
typedef std::string::const_iterator Cit;

bool contains(Cit b, Cit e, char c);
// PRE:  [b,e) ist ein gueltiger range
//       von Elementen des Typs char.
// POST: Rueckgabewert true <==>
//       c ist in *[b,e) enthalten.
```

Gruppe I

Implementieren Sie folgende Funktion:

```
typedef std::vector<unsigned int> Vec;
typedef Vec::const_iterator       Cit;

unsigned int even_sum(Cit b, Cit e);
// PRE:  [b,e) ist ein gueltiger range
//       von Elementen des Typs unsigned int.
// POST: Rueckgabewert ist die Summe
//       aller geraden Werte in *[b,e).
```

Gruppe J und K

Implementieren Sie folgende Funktion:

```
typedef std::vector<unsigned int> Vec;
typedef Vec::const_iterator       Cit;

unsigned int odd_sum(Cit b, Cit e);
// PRE:  [b,e) ist ein gueltiger range
//       von Elementen des Typs unsigned int.
// POST: Rueckgabewert ist die Summe
//       aller ungeraden Werte in *[b,e).
```

Institut für theoretische Informatik
Dr. B. Gärtner

25. Januar 2005

Informatik I: Material aus der Vorlesung

Programm: list.h

```
// Programm: list.h
// Doppelt verkettete Listen

namespace ifm {

  class ListElement {
  public:
    ListElement(int x, ListElement* p, ListElement* n);
    int data;
    friend class List;
  private:
    ListElement* prev_;
    ListElement* next_;
  };

  ListElement::ListElement(int x, ListElement* p, ListElement* n)
    : data(x), prev_(p), next_(n)
  {}

  class List {
  public:
    List(); // POST: Initialisiert zu leerer Liste.
    ~List(); // POST: Alle Listenelemente geloescht.

  private:
    // Kopieren verboten!
    List(const List&);
    List& operator=(const List&);

  public:
    void insert(int x, ListElement* i);
    // PRE: i ist aus [begin(),end())
    // POST: fuege x vor i ein.

    void erase(ListElement* i);
    // PRE: i ist aus [begin(), end())
    // POST: i aus Liste entfernt

  private:
    void destroy(ListElement* b, ListElement* e);
    // POST: alle Elemente aus [b,e) wurden geloescht,
    //   der belegte Speicher wurde freigegeben.

    ListElement sentinel;
  };

  List::List() : sentinel(0, 0, 0)
  {
    sentinel.next_ = &sentinel;
    sentinel.prev_ = &sentinel;
  }

  List::~List()
  {
    destroy(sentinel.next_, &sentinel);
  }

  void List::destroy(ListElement* b, ListElement* e)
  {
    while (b != e) {
      ListElement* d = b;
      b = b->next_;
      delete d;
    }
  }

  void List::insert(int x, ListElement* i)
  // PRE: i zeigt auf ein Element der Liste
  // POST: fuege x vor i ein.
  {
    ListElement* n = new ListElement(x, i->prev_, i);
    n->prev_->next_ = n;
    n->next_->prev_ = n;
  }

  void List::erase(ListElement* i)
  // PRE: i zeigt auf ein Element der Liste
  // POST: i aus Liste entfernt
  {
    i->prev_->next_ = i->next_;
    i->next_->prev_ = i->prev_;
    delete i;
  }

} // namespace ifm
```