Fachhochschule Aachen Fachbereich Elektrotechnik und Informationstechnik	**Klausur - Bachelor** Algorithmen und Datenstrukturen (52106) Prof. Ingrid Scholl	WS 2016/17 15.09.2015 Seite 1 / 17

Name (Blockschrift) Matrikelnummer

Hinweise:
Bearbeitungszeit: 180 Minuten
Hilfsmittel: Es sind keine Hilfsmittel zugelassen!

- Bearbeiten Sie die Aufgaben direkt und nur auf den Aufgabenblättern oder Leerbögen
- Lösungen ohne erkennbaren und leserlichen Lösungsweg sind wertlos.
- Klausur zusammenlassen!!!
- Deckblatt mit Namen und Matrikelnummer beschriften und unterschreiben!
- Die letzte Seite der Klausur beinhaltet eine STL-Hilfe.

Hiermit bestätige ich, dass ich die Hinweise zur Kenntnis genommen habe:

(Unterschrift)

Aufgabe	**Max. Pkt.**	**Punkte**	
1	**17**		
2	**18**		
3	**20**		
4	**20**		
5	**12**		
6	**10**		
Gesamt	**97**		**Note:**

Aufgabe 1 (17 Punkte)

(a) Wie viele Binärbaum-Formen mit N Knoten haben die Höhe N-1? Wie viele Möglichkeiten gibt es, um N verschiedene Schlüssel in einen anfänglich leeren binären Suchbaum einzufügen, der zu einem Baum der Höhe N-1 anwächst?

2  oder

(b) Nennen Sie die relevanten Komplexitätsklassen der O-Notation mit aufsteigender Komplexität.

$O(1) \rightarrow O(\log n) \rightarrow O(n)$ — .. siehe Vorlesung

(c) Welche O-Notation haben folgende Größen?

N+1 $O(N)$

1 + 1/N $O(1)$ weil $\lim_{N\to\infty} \frac{1}{1+\frac{1}{N}} = 1 < \infty$

log(2N) / log(N) $= \frac{(\log 2 + \log N)}{\log N} = \frac{1}{\log N} + 1 \Rightarrow O(1)$

$2N^3 - 15N^2 + N$ $O(N^3)$

log(N^2 + 1) / log(N) l'Hopital $\frac{2N}{N^2+1} \times \frac{N}{1} = \frac{2N^2}{N^2+1} \rightarrow O(1)$

$N^{100} / 2^N$ $O(0)$

(d) Wie lautet die Wachstumsordnung in O-Notation (als eine Funktion von N) der Ausführungszeit für das folgende Codefragment?

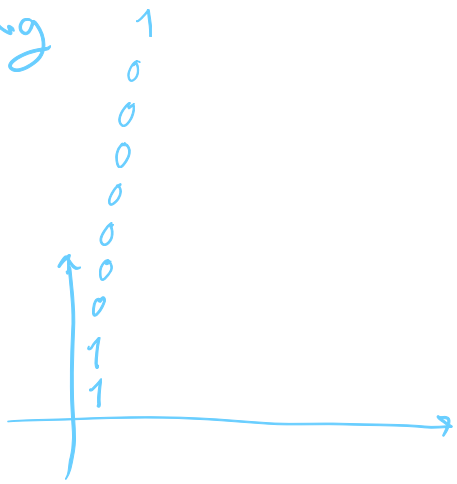
```
int sum = 0;
for (int n = N; n > 0; n /= 2)
   for (int i = 0; i < n; i++)
      sum++;
```

$N + \frac{N}{2} + \frac{N}{4} + \ldots = N\left(1 + \frac{1}{2} + \frac{1}{4} \ldots\right) = 2N \rightarrow O(N)$

(e) Was ist das Ergebnis der folgenden Implementierung für N = 1027? Zeigen Sie die einzelnen Stackzustände, dh. den Auf- und Abbau des Stacks sowie die Ausgabe.Was liefert der SourceCode als Ausgabe für eine beliebige positive Integerzahl N?

```
stack<int> s;
while (N > 0)
{   s.push(N%2);
    N = N/2;
}
while(!s.empty())
{   cout << s.top();
    s.pop();
}
cout << endl;
```

Binäre Darstellung



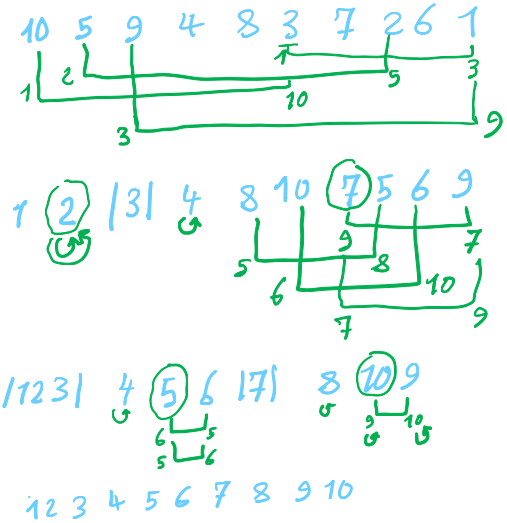
10000000 11

(f) Nennen Sie drei behandelte Sortieralgorithmen, die mit einem quadratischen Aufwand sortieren.

Siehe Vorlesung

(g) Gegeben sei die Folge: 10, 5, 9, 4, 8, 3, 7, 2, 6, 1. Zeigen Sie, wie die Folge mit dem QuickSort-Verfahren sortiert wird. Verwenden Sie als Pivot-Element stets das mittlere Element $\lfloor N/2 \rfloor$. Wie viele Vertauschungen mussten durchgeführt werden?

N=10



4 7 7 13

1 2 3 4 5 6 7 8 9 10

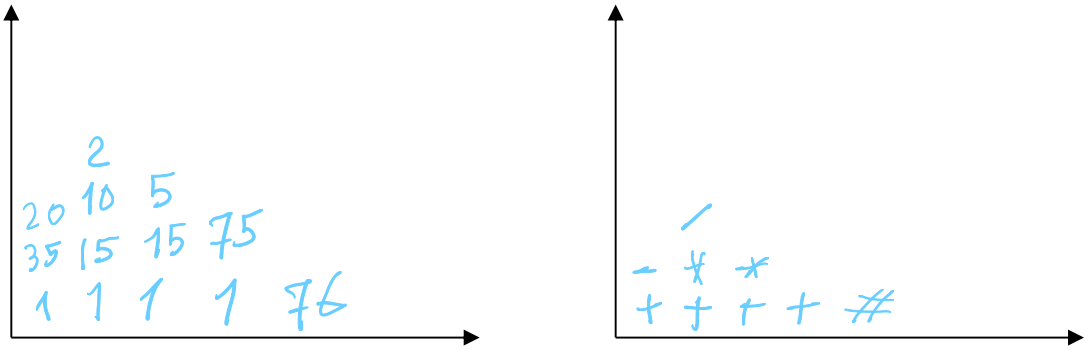
Aufgabe 2 (18 Punkte)
Auswertung arithmetischer Ausdrücke mit Stapeln

Implementieren Sie eine Methode, die einen arithmetischen Ausdruck korrekt berechnet. Der arithmetische Ausdruck sei als String gegeben und korrekt geklammert. Die Methode soll den String übergeben bekommen und das Ergebnis des Ausdrucks zurückgeben. (Beispiele zur Stringverarbeitung mit der STL finden Sie auf der letzten Seite. Die Eingabezeile selbst besteht aus einer Folge von Strings, dh. zwischen den Klammern, Operanden und Zahlen befindet sich jeweils ein Leerzeichen.)

Beispiel: (1 + ((35 - 20) * (10 / 2))) = 76

Verwenden Sie zur Lösung zwei Stapel. Im ersten Stapel legen Sie die Operanden ab und im zweiten Stapel die Operatoren. Öffnende Klammern werden übersprungen. Bei einer schließenden Klammer werden zwei Operanden vom Operandenstapel entnommen und ein Operator vom Operatorstapel. Die Operation wird berechnet und das Ergebnis wieder auf den Operandenstapel gelegt. Dies wird solange wiederholt, bis der Operatorstapel leer ist und im Operandenstapel nur noch 1 Element enthalten ist.

(a) Zeigen Sie die Zustände des Operanden- und Operator-Stapels zu obigem Beispiel.



(b) Implementieren Sie die Methode zur Auswertung. Verwenden Sie dazu die Datenstrukturen der STL. Gehen Sie davon aus, dass die Operanden und Operatoren jeweils durch ein Leerzeichen getrennt sind.

```
int Calculate (string A)
{   stack<int> operand;
    stack<char> operator;
    for(int i=0; i < A.lenght(); i++)
    {   char c= A[i];

        if( c==' ' || c =='(')
            Continue;
        if( c=='+' || c=='-' || c=='*' || c=='/')
            operator.push(c);

    else if( c==')'){
            int b = operand.top(); operand.Pop();
            int a= operand.top(); operand.Pop();
            char op=operator.top(); operator.Pop();
            int rs;
            if(op=='+')
                rs = a+b;
            if(op=='-')
                rs = a-b;
            if(op=='*')
                rs = a*b;
            else{
                assert(b!=0);
                rs = a/b;
            }
            operand.push(rs);
        }
        else{ // int einlesen
            string zahl = "";  int rs=0;
            while(A[i]!=' ')
                zahl += A[i];
            for(int j=zahl.length()-1, factor=1; j>=0; j--, factor*=10)
            {   rs += (zahl[j]-'0') * factor);
            }
            operand.push(rs);
    }   }
    return operand.top();
}
```

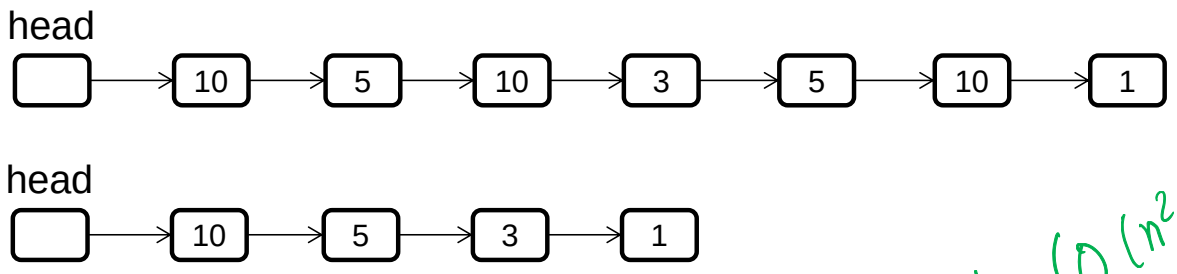
Aufgabe 3 (20 Punkte)
Entfernen von Duplikaten in einfach verketteten Listen

Gegeben sei eine einfach verkettete Liste. Die Knoten sind nach der Datenstruktur Node definiert.

```
class Node {
   int key;      // Schlüssel
   Item object;  // Datenobjekt
   Node* next;   // Referenz auf den Nachfolgerknoten
};
```

Implementieren Sie eine Methode, die alle Duplikate innerhalb einer einfach verketteten Liste erkennt, diese ausgibt und löscht. Begründen Sie, wie hoch der Aufwand in O-Notation zu ihrer Lösung ist.

Beispiel:
In der folgenden verketteten Liste wurden die Duplikate 10 und 5 erkannt und entfernt. Als Ergebnis erhält man die kürzere verkettete Liste ohne Duplikate.



Complexity $O(n^2)$
$1+2+3+\dots+n-1$

Ausgabe:
Folgende Duplikate wurden erkannt und entfernt: 10, 5

```
void List::removeDuplikate()
{   cout << "Folgende Dup... :";
for(Node* p = head->next; p!=nullptr && p->next!=nullptr; p=p->next)
{   Node *q_Prev = p;  bool duplicated = false;
    for(Node *q = p->next; q!=nullptr; q=q->next)
    {   if(p->key == q->key){
            duplicated = true; q_Prev->next = q->next;
            delete q;
            q = qprev;
        }
    } if(duplicated){cout << p->Key; if(p->next!=null)
                                        cout << ", ";}
}   }
```

Aufgabe 4 (10 + 10 Punkte)

(a) Welche ausgeglichenen Bäume kennen Sie?

(b) Warum werden Binärbäume ausgeglichen? damit die Suche O(log N) Komplexität hat

(c) Wann ist ein Binärbaum ein AVL-Baum?

Sei K ein Knoten ∈ AVL

$|\ K\to left\to height - K\to right\to height\ | < 2$

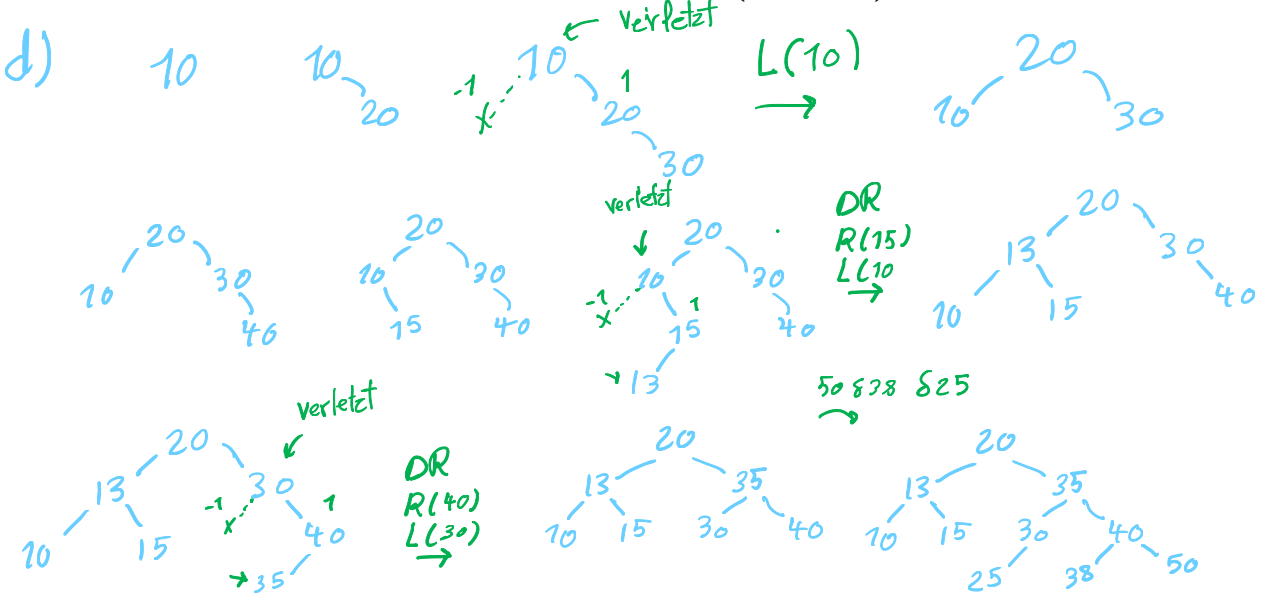
(d) Erzeugen Sie einen AVL-Baum aus der Folge {10, 20, 30, 40, 15, 13, 35, 50, 38, 25, 60} und zeigen Sie alle notwendigen Rotationen zur Herstellung des AVL-Baumes. Markieren Sie diejenigen Knoten, an denen das Balance-Kriterium verletzt wurde und benennen Sie die Rotation, um den Baum wieder auszugleichen.

(e) Implementieren Sie eine Methode, die überprüft, ob ein gegebener Binärbaum ein AVL-Baum ist. Die Knoten eines Binärbaums sind folgendermaßen definiert:

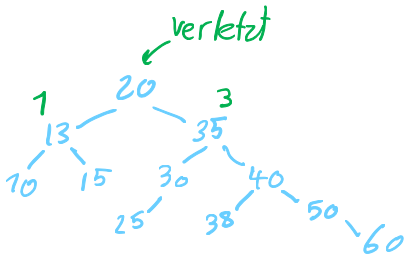
```
class Node{
   int key;
   Item object;
   Node* left;
   Node* right;
};
```

Der Binärbaum verfügt über einen Head-Knoten, der mit dem rechten Nachfolger auf den Wurzelknoten des Binärbaums referenziert. (10 Punkte)

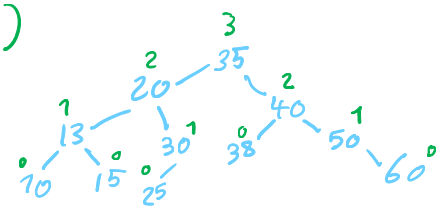
d)



Weiter auf 9



L(20)




```
//starter
bool AVL::Proof(){
    if(Pfoof(head->right)==-2)
        return false;
    return true;
}
int AVL::Proof(Node* n)   // gibt -2 zurück wenn Der Baum verkettet
{   if(!n)                //       Sonst dessen Höhe
        return -1;
    int L = Proof(n->left);
    int R = Proof(n->right);
    int B = L-R;
    if(-1 <= B && B <= 1 && L!=-2 && R!=-2)
        return (L>R)? 1+L : 1+R;

    return -2;
}
```

Aufgabe 5 (12 Punkte)
Hashing

Es soll eine Hashtabelle mit der Größe M gefüllt werden. Die Tabellengröße M sei immer eine Primzahl.

(a) Wozu verwendet man Hashing? Elemente speichern?

(b) Wie groß ist der Aufwand beim Einfügen und der Suche eines Elementes in einer Hashtabelle in O-Notation?

Suchen (O(1)) und Einfügen (O(1))

$3.45 \cdot 0.69 = 3.45$

(c) Die Zahlen **7, 15, 12, 2, 1, 20** sollen in eine anfangs leere Hashtabelle der Größe M=5 mit dem doppelten Hashing nacheinander eingefügt werden. Bei einem Belegungsfaktor > 0.69 soll ein Rehashing durchgeführt werden. Zeigen Sie den Zustand der Hashtabelle vor und nach dem Rehashing.

2

$\frac{x}{5} > 0.69 \Rightarrow x > 3.45$

Nur 3 Elemente

Vorher: $(x + i(3 - x\%3))\%5$

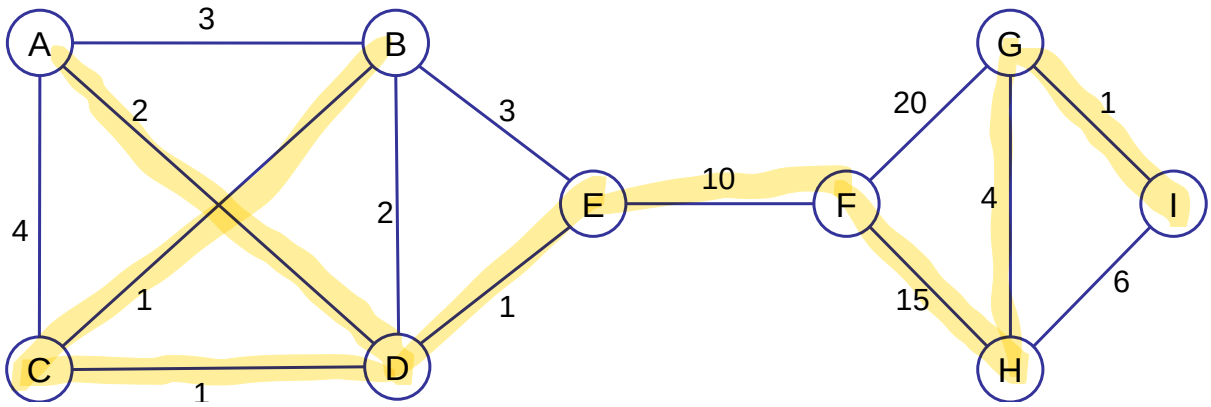
i	0	1	2	3	4
ht[i]	15	-1	7	12	-1

Nachher: $(x + i(7 - x\%7))\%11$

i	0	1	2	3	4	5	6	7	8	9	10				
ht[i]	-1	12	2	-1	15	-1	-1	7	-1	20	-1				

Aufgabe 6 (10 Punkte)

Gegeben ist der folgende gewichtete und ungerichtete Graph G = (V,E)



(a) Was ist der Unterschied zwischen Prim-Algorithmus und Dijkstra-Algorithmus?

Prim findet MST (Minimum Spanning tree) zum Startknoten
Dijkstra findet die kürzten Wege zum Startknoten

(b) Geben Sie die Adjazenzmatrix zu obigem Graphen an.

	A	B	C	D	E	F	G	H	I
A	0	3	4	2	0	0	0	0	0
B	3	0	1	2	3	0	0	0	0
C	4	1	0	1	0	0	0	0	0
D	2	2	1	0	1	0	0	0	0
E	0	0	0	0	0	10	0	0	0
F	0	0	0	0	0	0	20	15	0
G	0	0	0	0	0	20	0	4	1
H	0	0	0	0	0	15	4	0	6
I	0	0	0	0	0	0	1	6	0

(c) Berechnen Sie mit dem **Prim Algorithmus** den minimalen Spannbaum ausgehend vom **Knoten A**. Welche Gesamtkosten werden erreicht? Zeigen Sie die einzelnen Zustände der Priority Queue und markieren Sie die Updates, die in der Priority Queue durchgeführt werden mussten. Markieren Sie den Minimalen Spannbaum im Graphen.

Kante	-	AD	DC	CB	DE	EF	FH	HG	GI				
Kosten von A	0	2	1	1	1	10	15	4	1				
Knoten	A	D	C	B	E	F	H	G	I				

Zustände der Priority Queue:

Gesamtkosten: 35

AB3 AD2 AC4	E1 B2 C1	E1 B1	E1	F10	G20 H15	I6 G4	I1
D2	DC1	CB1	DE1	EF10	FH15	HG4	GE1

(d) Implementieren Sie eine Methode, die eine Kante aus dem Graphen entfernt. Als Eingabeparameter wird der Methode die Adjazenzmatrix als Call-by-Reference übergeben und ein Knotenpaar, das die zu entfernende Kante repräsentiert. Implementieren Sie die Knotennamen zur Vereinfachung als Integerzahlen (A = 0, B = 1, usw.).

```
Void removeKante (vector<vector<int>> &G, int A, int B)
{   int V = G.size();
    if( A<0 || A ≥ V || B < 0 || B ≥ V)
        return;

    G[A][B] = 0;
    G[B][A] = 0;

}
```

(e) Implementieren Sie eine Methode, die überprüft, ob der Graph nach Entfernung einer Kante in zwei Teilgraphen zerfällt. Entfernt man zum Beispiel in dem Graphen die Kante (E,F), dann entstehen zwei Teilgraphen. Verwenden Sie die rekursive Tiefensuche zur Traversierung, die Sie als gegeben voraussetzen können. Als Rückgabeparameter wird ein Vektor über alle Knoten zurück geliefert, wobei Werte = 0 nicht besuchte Knoten identifizieren.

```
vector<int> recursiveDepthSearch(map<int, vector<int> > adj,
                                 int startNode)
```

```
bool check(map<int, vector<int>> & adj)

{   int V = adj.size();
    vector<int> marked = recursiveDepthSearch(adj, 0);
    for(int i=0 i<V;     i++)
        if(marked[i] == 0 )
            return false;

    return true;
}
```

STL-Hilfe:

Container, einige wichtige Funktionen und deren Verwendung:

vector<type> Feld

```
vector<int> numbers;
numbers.push_back(3);        // Einfügen am Ende des Feldes
numbers.pop_back();          // entfernt das letzte Element
int last = numbers.back();   // liefert das letzte Element
int first = numbers.front(); // liefert das erste Element
int item = numbers[i];       // liefert das i-te Element
int size = numbers.size();   // liefert die Größe des Feldes
bool empty = numbers.empty(); // true, wenn das Feld leer ist,
                              // andernfalls false
```

list<type> doppelt verkettete Liste (s.Funktionen wie bei vector)

```
list<int> numbers;
numbers.push_back(5); // fügt 5 ans Ende der Liste
numbers.remove(5);    // löscht das Element 5
numbers.push_front(1); // fügt 1 am Kopf der Liste ein
```

queue<type> Queue

```
queue<int> q; // int – Warteschlange allokieren
q.push(5);    // Element 5 in die Warteschlange
bool leer = q.empty();   // Abfrage, ob Warteschlange leer ist
int element = q.front(); // liefert erstes Element
q.pop() ;     // löscht erstes Element
```

stack<type> Stapel

```
stack<int> nodes; // int - Stapel allokieren
bool ok = nodes.empty();   // überprüfe, ob Stapel leer ist
int element = nodes.top(); // hole Element vom Stapel
nodes.pop();   // entfernt Element vom Stapel
nodes.push(5); // 5 wird auf den Stapel gelegt
int i = (int) nodes.size(); // Anzahl Elemente im Stapel
```

map<keytype, valuetype> (Schlüssel, Referenz)-Container

```
map<int, vector<int> > adjlist;    // Adjazenzliste als Map
int size_adjlist = adjlist.size(); // Anzahl Knoten in der Adjazenzliste
vector<int> liste = adjlist[1]; // liefert den Vektor zum Schlüssel 1
adjlist[1].push_back(5); // Einfügen von Schlüssel 5 zur Liste von Schlüssel 1
```

Beispiel zur Verwendung von Strings mit der STL:

```
// string to double example
#include <iostream>   // std::cout
#include <string>     // std::string, std::stod

int main ()
{
  std::string orbits ("365.24 29.53");
  std::string::size_type sz;     // alias of size_t

  double earth = std::stod (orbits,&sz);
  double moon = std::stod (orbits.substr(sz));
  std::cout << "The moon completes " << (earth/moon) << " orbits per Earth
year.\n";
  return 0;
}
```

```
// string::begin/end
#include <iostream>
#include <string>

int main ()
{
  std::string str ("Test string");
  for ( std::string::iterator it=str.begin(); it!=str.end(); ++it)
    std::cout << *it;
  std::cout << '\n';

  return 0;
}
```

Output: Test string

```
// comparing apples with apples
#include <iostream>
#include <string>

int main ()
{
  std::string str1 ("green apple");
  std::string str2 ("red apple");

  if (str1.compare(str2) != 0)
    std::cout << str1 << " is not " << str2 << '\n';

  if (str1.compare(6,5,"apple") == 0)
    std::cout << "still, " << str1 << " is an apple\n";

  if (str2.compare(str2.size()-5,5,"apple") == 0)
    std::cout << "and " << str2 << " is also an apple\n";

  if (str1.compare(6,5,str2,4,5) == 0)
    std::cout << "therefore, both are apples\n";

  return 0;
}
```