Algorithmen und Datenstrukturen
Aufgabenblatt 1
Entwurf von Datentypen

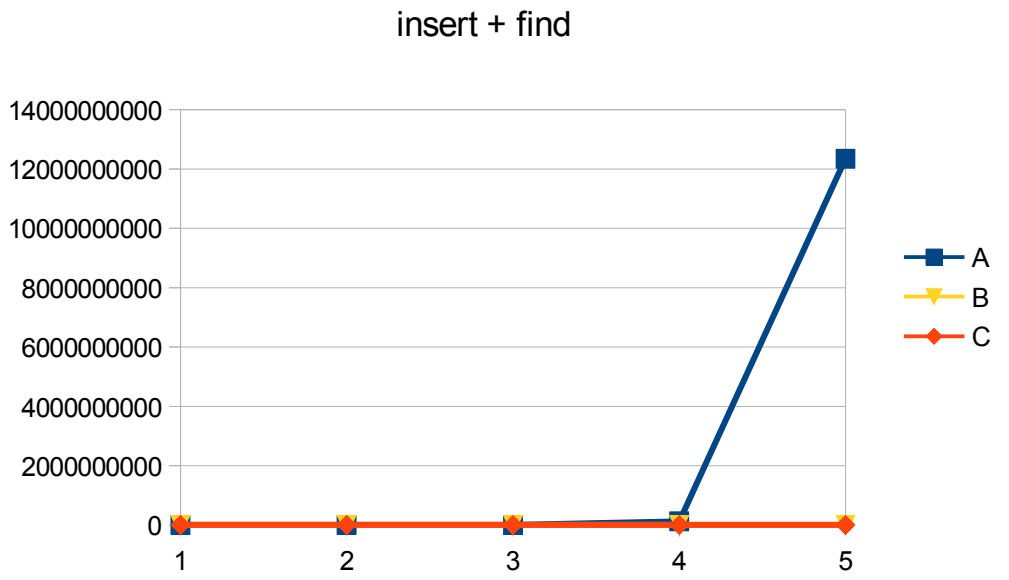
Johannes Kruber 2288692
Luis Nickel 2199554
Felix Naumann 2210645

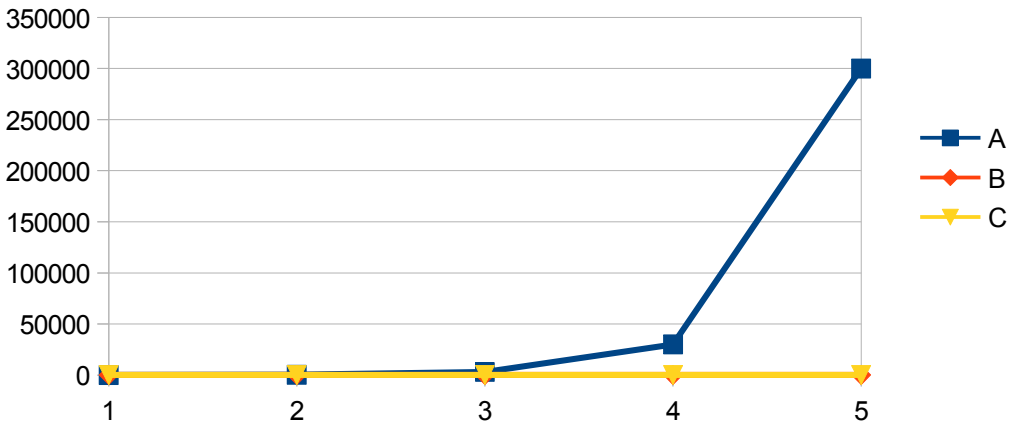
1. Pre- and Postconditions:

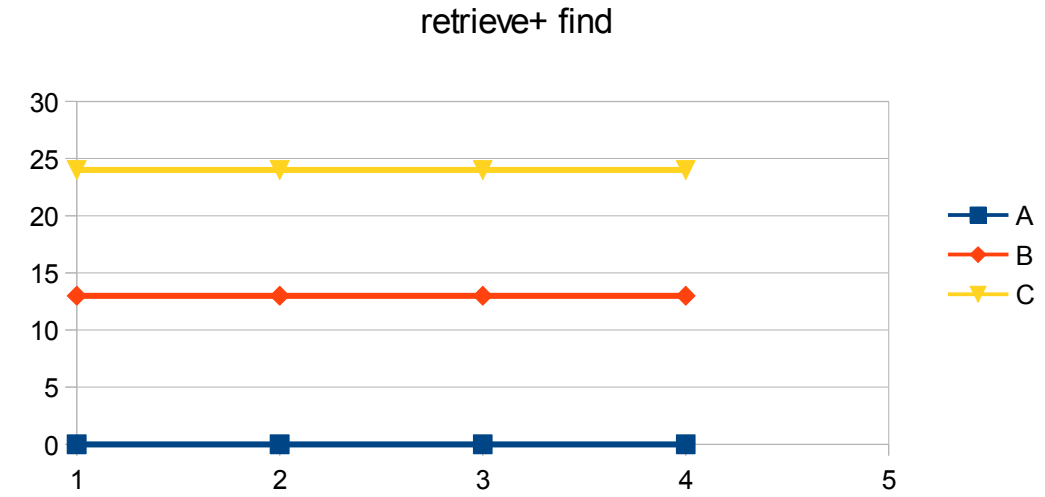
	A	B	C
insert	Pre: pos € int pos inerhalb der array elem € T Post: insert(p,e) = (x1 …... , e, ….....xn) p € int , e € T	Pre: pos € ElementB<T> pos inerhalb der array elem € T Post: insert(p,e) = (x1 …... , e, ….....xn) p € ElementB<T> , e € T	Pre: pos € ElementC<T> pos inerhalb der array elem € T Post: insert(p,e) = (x1 …... , e, ….....xn) p € ElementC<T> , e € T
delete	Pre: pos € int pos inerhalb der array post: delete(p) p € int	Pre: pos € ElementB<T> pos inerhalb der array post: delete(p) p € ElementB<T>	Pre: pos € ElementC<T> pos inerhalb der array post: delete(p) p € ElementC<T>
delete	Pre: k € String Post: delete(k) k € String	Pre: k € String Post: delete(k) k € String	Pre: k € String Post: delete(k) k € String
find	Pre: k € String Post: find(k) k € String, e € T	Pre: k € String Post: find(k) k € String, e € T	Pre: k € String Post: find(k) k € String, e € T
retrieve	Pre: pos € int pos inerhalb der array post: retrieve(p)= (x1...,e,...xn) e € T, p € int	Pre: pos € ElementB<T> pos inerhalb der array post: retrieve(p)= (x1...,e,...xn) e € T, p € ElementB<T>	Pre: pos € ElementC<T> pos inerhalb der array post: retrieve(p)= (x1...,e,...xn) e € T, p € ElementC<T>
concat	Pre: l € ListeA Post:concat(l) = (elemente des array , elemente von l) l € Ilist	Pre: l € ListeA Post:concat(l) = (elemente des array , elemente von l) l € Ilist	Pre: l € ListeA Post:concat(l) = (elemente des array , elemente von l) l € IList

Das UML-Diagramm der Software befindet sich im Anhang.
Die Funktionssignaturen sind dem UML-Diagramm zu entnehmen.

2.
In den nachfolgenden Diagrammen ist die Aufwandsanalyse zur implementierten Software zu sehen. Hier bei sind auf der Y-Achse die Anzahl der Operationen aufgeführt und auf der X-Achse k, wobei die Anzahl der Elemente in der Liste 10^k entspricht.
Es wurde immer die entsprechende Funktion und die find-Funktion benutzt, um einen brauchbaren pos wert zu erhalten. Auch wurde immer auf das 4 Element zugegriffen.
Es sind alle Implementierungen innerhalb eines Diagrammes zu sehen, um diese zu vergleichen. Man erkennt das bei der Implementierung der ListeA die Operationen stark zunehmen, während bei den anderen Listen die Anzahl konstant bleibt. Eine Ausnahme bildet die retrieve-Funktion, hier sind alle Implementierungen konstant.
Die Anzahl der Operationen steigt bei allen Listen in Abhängigkeit zur Position des Elementes auf das zugegriffen wird, je weiter hinten das Element in der liste liegt desto höher ist die Anzahl der nötigen Operationen.


insert + find
14000000000
12000000000
10000000000
8000000000
6000000000
4000000000
2000000000
0
1
2
3
4
5
A
B
C


delete + find
350000
300000
250000
200000
150000
100000
50000
0
1
2
3
4
5
A
B
C


retrieve+ find
30
25
20
15
10
5
0
1
2
3
4
5
A
B
C