**Lab 1: Exercise 2**

Estimated worst-case time complexity (Big-Oh) of a number of functions

S has *n* elements

S1 has *n* elements

S2 has *m* elements

## Set S1(S2);

A node pointer is created = **1** step

An outer for-loop over all elements in S2 = **m** steps

Each iteration of the for-loop does an insertion which is a constant number of steps **k**

T(m) = 1 + k \* m = O(m)

## S1 = S2;

A temporary set is constructed by calling the copy constructor on S2: **1 + k \* m** steps

The temporary set is swapped with S1. The std::swap has a time complexity of **O(1)** according to cppreference.com

Finally the temporary set (that now holds the n original elements from S1) is destroyed. The destructor needs to destroy all n elements: **k2 \* n** steps

T(m, n) = (1 + k \* m) + (k2 \* n) + O(1) = O(n + m) // still linear time complexity

## S.cardinality();

Returns the counter value: **1** step

T(n) = 1 = O(1)

## S1 + S2

A temporary set is created by calling the copy constructor on S1: **1 + k1 \* n** steps

The *+=* operator is called with the temporary set as “this” and S2 as argument:

* The method has a constant number of steps outside of its while-loop: **k2**
* An outer while-loop *L1* iterates over all elements in S2: **m** iterations
* Each iteration has a varying number of steps, depending on input. Each iteration performs at least a constant number of steps: **k3**. An inner while-loop *L2* performs, over the course of all L1-iterations, in total **k4 \* n** steps by iterating over all elements in S1.

The operator+ method calls the destructor of the sum set: **k5 \* (n + m)** steps

T(n, m) = (1 + k1 \* n) + k2 + (m \* k3) + (k4 \* n) + (k5 \* (n + m)) = O(n + m) // still linear time complexity

## k+S;

Converts k to a set and inserts k as a value, a constant number of steps **k1**.

Calls the same method as above (i.e. Sk + S) which is **O(m + n) = O(n)** complexity (since m = 1 in this case)

T(n) = k1 + O(n) = O(n)