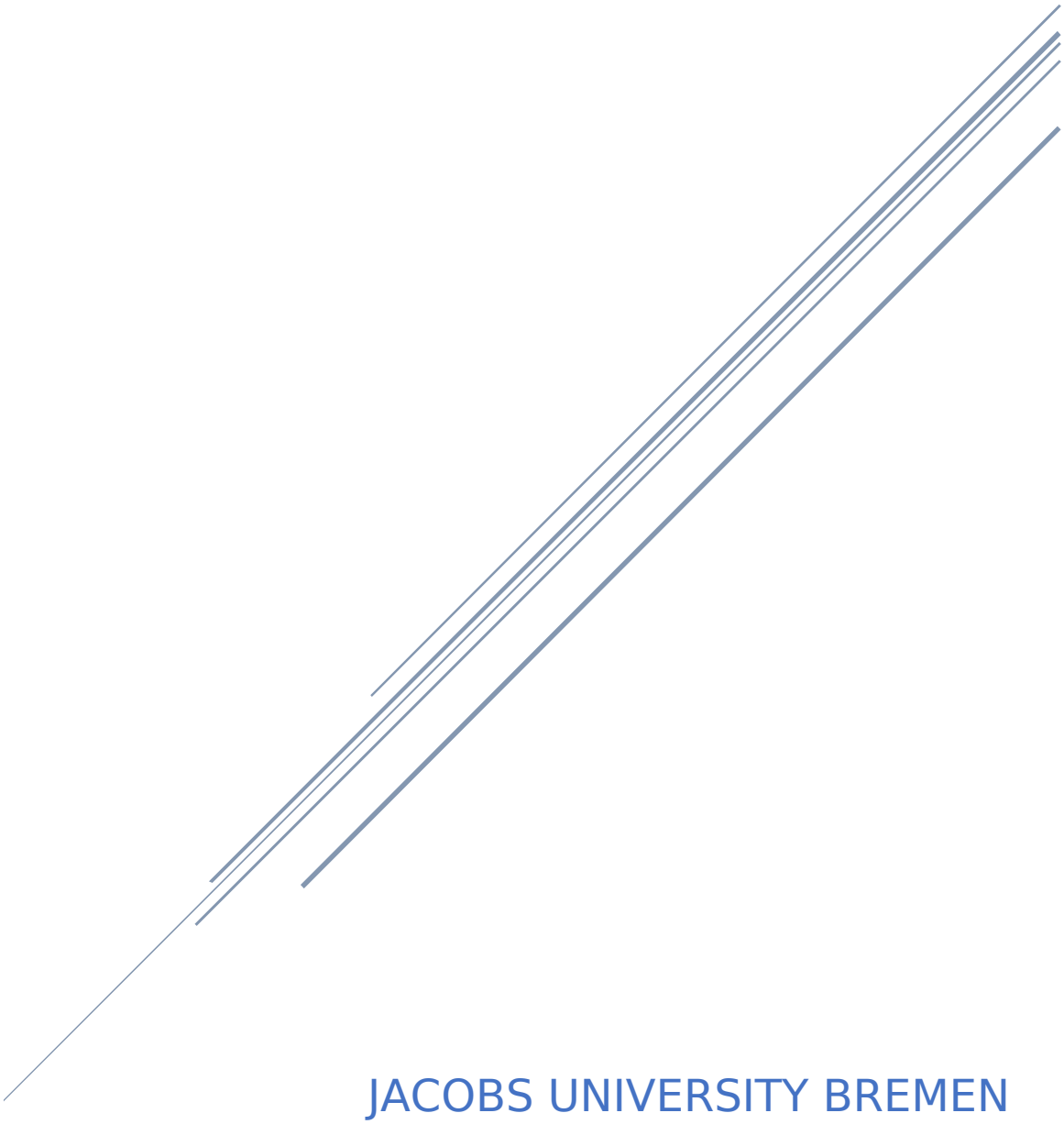
# HOMEWORK 8

## ALGORITHMS AND DATA STRUCTURES



JACOBS UNIVERSITY BREMEN
Mario Alberto Hernandez Salamanca

# Problem 8.1

a)

The time complexity for each of the functions is Θ(1) because they are constant functions, we don't go through the all stack we just add at the top and delete add the top or we just check if there is at least 1 element.

## Problem 8.2

```
T inverse_list(list<T> mylist){
        struct list* next = NULL
        struct list* current = mylist->head
       struct list* before = NULL
       while(current != NULL){
        next=current->next;
       current->next = before;
        before=current;
        }
        mylist->head = before;
}
```

b)

Time complexity of the algorithm is O(w*n) where w is width of Binary Tree and n is number of nodes in Binary Tree. In worst case, the value of w can be O(n) (consider a complete tree for example) and time complexity can become $O(n^2)$.

C)

Time complexity of the algorithm is O(n) because you are going to move between the whole list so depending on the size of the list that is how the complexity is going to growth.