Report of TPK4186 assignment 1

**Creators - Group 32:** Martin Kristiansen Tømt and Nikolay Westengen

**Modules**
This assignment contains the following modules:

containerlist.py
Containers.py
Ships.py
docks.py
Main.py
Visualisering.py (This one has been manipulated to create a the different pictures, therefore it is added here for so the reader can see principles of it)

## Tasks

## 2.1 Containers
The structures and functions used in these tasks can be found under Containers.py and countainerlist.py.
An example run of this code can be found in main.

**Task 1**
The structure can be found in the Containers.py script.

**Task 2**
The structure can be found in containerlist.py

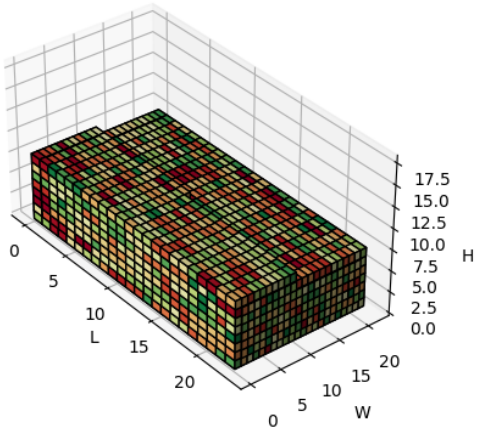
**Task 3**
This function can be found i containerlist.py

**Task 4**
This function can be found in containerlist.py. If the file doesn't exist from before, it will create a file for it

## 2.2 Ships
The structures and functions used in these tasks can be found under Ships.py
An example run of these functions can be found in main.

**Task 5**
**Task 6**

**Task 7**

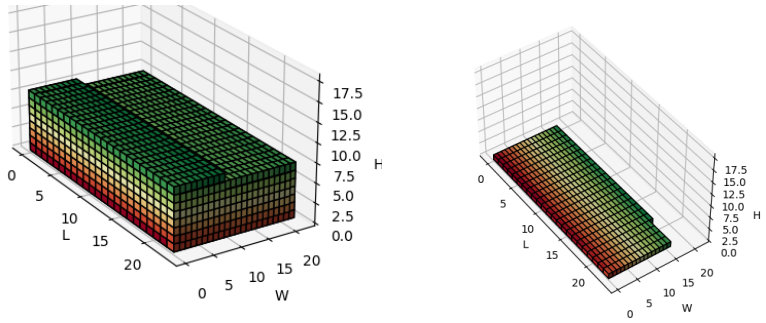



This is a 3d plot of the containers; green = light, red = heavy. One block equals one TEU, so a 40 foot container fills two blocks. This is the loadContainer() function and it has in this situation not been used on a list that is sorted, therefore the position of the containers are random and does not care about weight.

```
loadShipWithContainerList
```

This line sorted the containers in descending weights.

**Task 8**



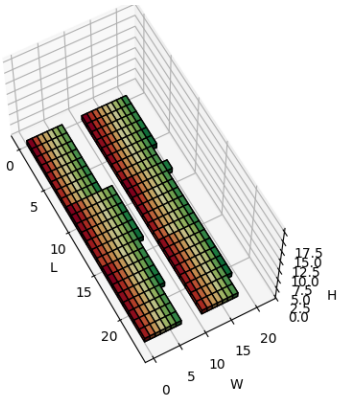

This is a 3d plot of the containers, with the sorted algorithm. As one can see, the algorithm always starts a new layer in one corner, which can offset the balance.

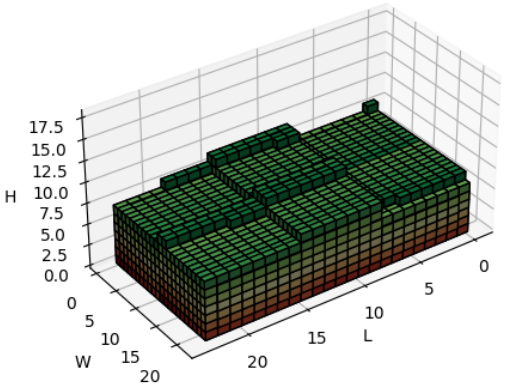
**Task 9**

The isShipBalanaced() function does not take into account the first 750 tonnes before caring about sections, this is just an assumption we took because we regarded it as safe, just to make the function a bit more relevant and convenient.

**Task 10**

This is the function that takes weight distribution into account, and loads each of the sections in a way that does not offset the balance too much.




```
findAvailableContainerSpot
```

This line changed it so we distribute the weight properly

## 2.3 Ships

The structures and functions used in these tasks can be found under docks.py.

**Task 11**

There are two functions here, one that counts the containers on the docks and one that counts the containers loaded on the ship. This gives both a function for loading and unloading time.

**Task 12**

The function has a priority system bay 1 - bay 2 - bay 3 - bay 4, where bay 1 is most prioritized and bay 4 is the least prioritized. we wanted crane one to choose the bay to remove a container from first, then cranetwo, while avoiding crane 1, then crane 3 avoiding crane 2 and finally crane 4 while avoiding crane 3. This would potentially decreased the time as far as three fourths from 1 crane. However, if all the containers was within the reach of just one crane, there would be no difference