Exercise 1

Fredrik Faanes Slagsvold

This program is made in accordance with the tasks that were given in the document explaining the exercise. As a result the main functionalities of this program is to generate a set of containers, generate a ship and load the containers on this ship. A set of containers can be saved to a file and also loaded from a file. The same goes for the load of a ship. This can be saved and loaded onto another ship. Furthermore there are many other functions that implements other types of functionalities. These were also made according to the tasks given in the exercise. All the functions have been tested thoroughly in order to make sure that the work the way they are intended. In some of the tasks assumptions have been made in order to fulfill all of the suggested implementations.

I chose to write the whole program in one file because that was the way it was done in the lectures. However i will do it fully object oriented with differenct classes next time because this gives a much better overview of the code and the different functions.

The main function consists of code that is supposed to show what the ship looks like after it has been fully loaded, and how long it takes to load and unload the ship.

I have made a section consisting of tests. Each of these have to be commented in in order to run. The tests were commented out so that they would not interfere with the output of the code in the main function.

Assumptions:

- Containers are loaded lengthwise down the ship

- If a 20-foot container is to be placed on a 40-foot container, it must be part of a set of two containers

- Containers cannot be sorted in advance. When loading the ship, one container should be considered at a time. If a container does not match what is desired to be loaded, it can be set

aside. For example, one 20-foot container can be held back until another 20-foot container is obtained.

- All ships are of the same size: length 24, width 22 and height 18

- A ship object is a list containing length, width, height and the load.

- It is possible to have 12 40-foot containers lengthwise on the entire ship.
12/3 = 4 containers lengthwise per section (20-foot container takes up half a space). 22/2 = 11 containers widthwise.

- It is possible to have 18 containers heightwise. This means that each stack will have a length of 18.

- The ship is divided into 6 sections: front, middle, and back, with starboard and port on each side. Each section is its own list. Each section contains 4*11 stacks. Each stack has room for 18 containers. A pair of 20-containers counts as one.

- 20-foot containers are paired together to prevent gaps. At the same time, it means that if there is an odd number of containers in the set, the last 20-foot container will not be loaded onto the ship.

- It is assumed that the weight distribution of containers is not critical when the first containers are loaded due to the ship's own weight. The ship will not meet the balance requirements when the first containers are loaded if the ship's own weight is not taken into account. After around 150 containers, the balance requirements will be met.

- It is assumed that 20-foot containers can be loaded at the same time (in pairs). This means that when moving a pair of 20-foot containers to place a heavier container, it counts as 1 operation.

- It is assumed that 2 20-foot containers can be unloaded from the ship at the same time (in pairs). This means that when unloading 2 20-foot containers, it counts as 1 operation.

RemoveContainer: Takes into account that there cannot be gaps by removing the entire pair of 20-foot containers even though only one of the two is to be removed.

- LoadSingleContainer: Does not take into account that 20-foot containers should be loaded in pairs. In other words, it does not take into account gaps.

- Alternative assumption to shorten the time it takes to load the ship full with one crane: it takes 4 minutes from when a container is to be loaded until it has been loaded and the associated stack has been sorted.

- Task 12: Assumption: there are three cranes instead of four because this is more in line with the way I have chosen to structure my ship. Observation: The time it takes to fully load and unload the ship is one third of the time it took when using one crane.

- Ship_loadCargo(): This function is made in order to fulfill the requirements from all the loading-tasks in the exercise. In order to make sure that the ship stays in balance when loading the program finds the lightest section and the lighest stack every time a container is loaded. This is done in the loadSingleContainer()-function. Every placement and shuffle of a container in a stack is considered an operation, and the function counts each operation in each section of the ship: front, middle and back. the loadCargo()-function uses the PlaceContainerAndSortStack()-function. This takes in a container and the stack it is supposed to be placed in and compares the weight of the container to every container in the stack. Every container in the stack lighter than the new container has to be removed before the new one is placed after that it can be placed back.

- Ship_RemoveContainerFromShip(): When removing a container from a ship one has to find it in the correct stack. After this is done all the containers that are placed on top of the chosen container are popped and put back.

- Ship_UnloadShip(): searches for the heaviest section and stack in the heaviest section for every container that is popped. This is done in order to fulfill the requirements regarding balance.