Assignment 1

Mother-daugther ship representation (1-a)

```
example = [6,[5],[7,8],4,[3,2,1]]
```

My solution representation is a list containing either intergers or another list.
Every port is only in the list 1 time each.
Only restriction is that the first element of the list is a interger, as the mothership need to dock in atleast one port.

Example explanation

In the `example` above the mother ship will dock at port 6 send out 2 daugther ships, one to port 5, and another to the ports 7 and 8. Then the mother ship will dock at port 4 and send out a single daughter ship that goes to port 3, 2 and 1.

Changing solution

Changing a solution to another is easy with the representation.

- You can swap any two numbers.
- You can add/remove a list around a number (assuming its not already in a list).
- You can move a number out of a sublist to its own sublist or just to the main list.

With these 3 opperations all possible solutions is possible to make.
Just make sure that the first element is a integer.

Validating solution

Validating is also simple.
Go through the list and make sure the first element is a interger and that every port is in the list exactly once.

Readability

This is atleast for that small example easy to read and understand, if the mothership sends outs alot of daugtherships then it could be hard for a human to see where the origin port is.
Also very easy for a computer to read.

Compact

It is a two dimentional representations, but sense every port is at most 1 time its still as compact as can be, atleast regards to the data.

Could have many sublists that would take some space, but you would need to make some kind of differentiation anyway.

Solution space

If you lock the first element to be a port(integer) then the solution space is one to one.
So lets keep is that way :)

Truck-drone representation (1-b)

```
truck =       [0, 10, 9, 8, 7, 3, 5, 6]
blue_drone =  [(0,11,9), (3,1,5)]
black_drone = [(10,12,9), (7,4,3), (3,2,5)]
```

The truck list is the "main" nodes the truck goes to in order.
the lists for the drones are in the order the drones go out to deliver and each delivery is on the format: (X, Y, Z) where
X is the node the drone departs from the truck
Y is the node the drone is delivering to
Z is the node the drone meets up with the truck again

Validating solution

Checking the solution can be done with checking if every number is in either the truck list, or in the middle of the tuples of either of the drone lists.
You also need to check that the first and last number in the tuples are in an the same order as the truck list.
Lastly you need to check that one drone is not going on two jobs at the same time, do this by checking every last number in a tuple with the first of the next one.

Changing solution

Changing the solution can be done, but is not very easy.

Readability

It is very easy to read and understand what is happening with this representation.

Compact

It is fairly compact as in the worst case is will only store 3 times the number of nodes.