1. **Graph exercise 1** 
   1. ***Representation information in a graph and using graph algorithm to solve problems / calculate answers:***

The bus schedule (appendix) can be used to create a graph representing the time used to drive between the cities and the possibly routes.

* 1. ***Task 1.1:***

Draw a directed weighted graph for the route plan (using paper and pencil) with the driving time as the cost.

* 1. ***Task 1.2:***

In this part you don’t need a name for the cities – they might be represented just with an index.

You must now make a class containing the graph structure (Adjacency Matrix) that can represent the graph with methods for fill in data for vertices and edges. You might have a method for adding a directed edge and also one for adding an undirected (just duplicate for both direction).

Make a method (PrintMatrix) that write out the matrix for simple manual control.

Make a simple program that can fill in the graph with the route plan (directed graph) from Task 1.1 and use the PrintMatrix method to check the graph.

Expand the program with a with another graph object where you fill in the data but this time as undirected (both direction) and test also this using the PrintMatrix.

* 1. ***Task 1.3:***

Program a TraversDeepFirstRecursive method that print the cities in the order they are visited using deep first algorithm. A city must only be printed ones and the method must end. Start city must be given as argument for the method (for now it is ok just using the number (index) for the city and not the name.

Test the method with different start city and both the undirected and the directed version.

* 1. ***Task 1.4:***

Program a TraversBreadthFirst method that print the cities in the order they are visited using breath first algorithm. A city must only be printed ones and the method must end. Start city must be given as argument for the method (for now it is ok just using the number (index) for the city and not the name.

Test the method with different start city and both the undirected and the directed version.

* 1. ***Task 1.5:***

Program a TraversDeepFirst doing the same as task 1.3 but without using recursion.

Test the method with different start city and both the undirected and the directed version.

***Task 1.6:***

Expand the solution so when traversing you can print (or return a list with) the route used (city from - city to) and not only the city visited, still only the number /index for the cities is needed.

Test the method with different start city and both the undirected and the directed version.

***Task 1.7:***

Program a new version of the solution where the graph can contain the names of the cities and you can write the name of the city when traversing and use city name as argument when traversing.

Test the method with different start city and both the undirected and the directed version.

***Extra Task for the hardcore students.***

Make a solution with a graph using Adjacency List instead of the Adjacency Matrix

**Appendix**

Bus schedule / route plan (partial) on Fyn.

#44

odense 14:00

middelfart 15:07

#72

bogense 12:39

otterup 13:46

odense 14:08

#531

ringe 08:41

assens 09:24

odense 10:23

kerteminde 10:56

#62

svendborg 07:45

odense 09:15

bogense 10:04

#844

bogense 09:08

middelfart 09:46

assens 10:47

#22

svendborg 13:31

nyborg 14:16

odense 15:14

#32

svendborg 12:39

ringe 13:35