Lukas Sjöstrand and Malte Kasemo reporting on Simon Rödén's Lab 3

Section 1:
Q1: yes
Q2: yes
Q3: yes
Section 2:
B1: no

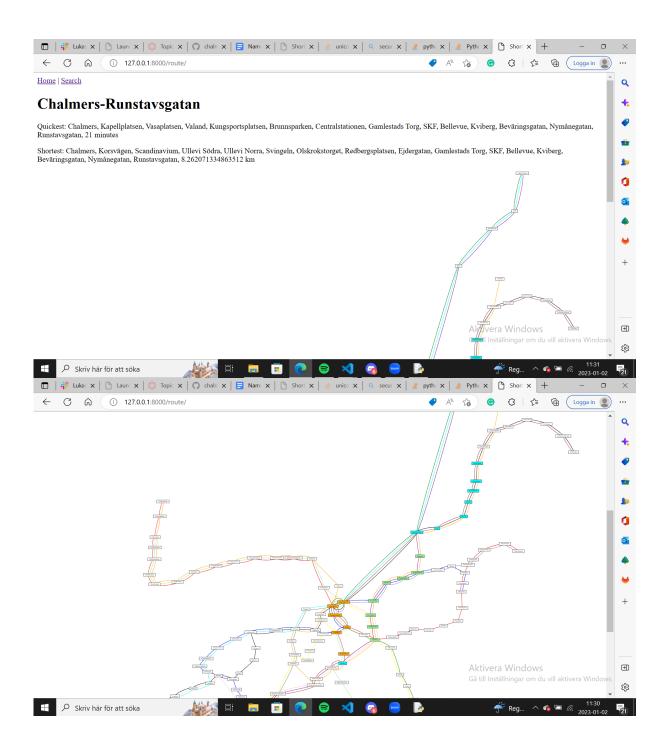
Section 3:

B2: yes

The code from lab 2 has been properly reused and dijkstra has been implemented and used as intended, where there is just one definition of the function. Just the cost function that have changed.

Section 4:

screenshot 1



Screenshot 2

```
from django.conf import settings
import numpy as mp
def show_shortest(dep, dest):
    network = readTramNetwork()
    quickest = dijkstra(network, dep, None)[dest]
        if stop in network._timedict:
             if quickest[idx+1] in network._timedict[stop]:
    time += network._timedict[stop][quickest[idx+1]]
                  time += network._timedict[quickest[idx+1]][stop]
             time += network._timedict[quickest[idx+1]][stop]
    def distance_between_stops(somedicts, stop1, stop2):
         data-somedicts
        phi1=lat1 = np.pi/180*float(data[stop1]["lat"])
phi2=lat2 = np.pi/180*float(data[stop2]["lat"])
         r= 6371
         lambda1=lon1 = np.p1/180*float(data[stop1]["lon"])
         lambda2=lon2 = np.pi/180*float(data[stop2]["lon"])
        a = (np.square(np.sin((phi2-phi1)/2)) + np.cos(phi1) * np.cos(phi2) *
            np.square(np.sin((lambda2-lambda1)/2)))
         d = 2*r*np.arcsin(np.sqrt(a))
         return(d)
    shortest =dijkstra(network, dep, cost=lambda u,v: distance_between_stops(network._stopdict,u, v))[dest]
     for idx, i in enumerate(shortest[:-1]):
         distance += distance_between_stops(network._stopdict, i, shortest[idx+1])
    timepath = 'Quickest: ' + ', '.join(quickest) + ', '+ str(time) + ' minutes'
geopath = 'Shortest: ' + ', '.join(shortest) + ', ' + str(distance)+' km'
    def colors(v):
         if v in shortest and v in quickest:
         elif v in shortest:
             return 'lightgreen'
         elif v in quickest:
           return 'white'
  ners-advanced-python-main updated 〉 lab3 〉 tram 〉 utils 〉 🍖 tramviz.py
```

```
圖 Pr... ② 王 ÷ ‡ 中 《 color_tram_svg.py × 《《 tramviz.py × 《 graphs.py × 《 farams.py × M chalmers-advanced-pyth 1
  > examples
> exercises
> lab3
                                                 network = readTramNetwork()
time = 1000000
for edge in network.sl_vertices():
              # __init__.py
              🖧 settings.py
                                                             a = edge[8]

if a == dest:

new_time = dijkstra(network, dep, cost=lambda u, v: network.get_weight(u, v))[edge]['cost']
              urls.py
          myvenv
bin
include
bilib

static
tram
Dimigrations
templates
Tram
                                                              a = edge[0]
if a == dest:
                                                                    new_dist = dijkstra(network, dep, cost=lambda u, v: network.geo_distance(u, v))[edge]['cost']
if new_dist < dist:
             images
ggg_tran
gshortest
find_route.h
home.html
show_route.
                                                                              dist = new_dist
                                                          quickest_path = path_time
quickest_time = time
                                                          shortest_lame = lime
shortest_dist = dist
timepath = 'Quickest: ' + ', '.join(quickest_path) + ', ' + str(quickest_time)
geopath = 'Shortest: ' + ', '.join(shortest_path) + ', ' + str(shortest_dist)
                機 __init__.py
機 color_tram_
機 graphs.py
機 trams.py
                                                           def colors(v):
```