Optimization

SETUP GO
TO install Go
Add the Go path and Go root in the .bashrc file
-> Go root will be the path installed go lang package
->Go path is working director of your Project.Ex.Mahindra

Basics Needed:
->Functions,variables,Go routines
->Maps,Structs,Concurrency,Testing
->Pointers,Loops,HTTP handlers,Error handling,Mutex
OSRM setup:
-> download this and put in an folder named osrm change name of this downloaded file as osrm-backend
-> run this following commads in the folder where osrm-backend folder is present

```
$ sudo apt install build-essential git cmake pkg-config \
libbz2-dev libxml2-dev libzip-dev libboost-all-dev \
lua5.2 liblua5.2-dev libtbb-dev
$mkdir -p build
$ cd build
$ cmake ..
$ cmake --build .
$ sudo cmake --build . --target install
```

-> cd to folder that contains folder of osrm run following

```
->install docker
$ sudo usermod -aG docker $USER
$ wget http://download.geofabrik.de/asia/india-latest.osm.pbf
$ docker run -t -v "${PWD}:/data" osrm/osrm-backend osrm-extract -p /opt/car.lua /data/india-latest.osm.pbf
$docker run -t -v "${PWD}:/data" osrm/osrm-backend osrm-partition /data/india-latest.osrm
$docker run -t -v "${PWD}:/data" osrm/osrm-backend osrm-customize /data/india-latest.osrm
```

-> run the osrm on local

```
$docker run -t -i -p 5000:5000 -v "${PWD}:/data" osrm/osrm-backend osrm-routed --algorithm mld /data/india-latest.osrm
```

Run the code:

Run the make file or bash file

Code-1 Car:

->defined two type trip,driveop with respective fields

->Input is in the format of a.csv,parse1() takes the csv and parses the trip into tripsbyship that map the trip to there time of day with interval of 30 min

->for each element of tripbyshift we run the function of findavailabledriver to assign each shift that slice of trips a driver

-> Here we are using go routine to assign drivers concurrently for all trips of that shift(30 min shifts)

-> For each trip,finding if there is any driver near by,and if they can reach the trip location in time,we assign that trip to that driver else,assign it to a newdriver and push him into the drivers slice

-> this outputs the csvtrips that which driver is assigned to which trip,and also cssvdrivers that cotaines drivers detailes the end trip status and all the trips they were assigned

->we have python code to get the graph from csvtrips

Code-2:(Still can be optimized needed data)

->mostly using the clustering code of proximity code in moove-project

->Here defining three type cluster,vehicle,packet(load) that cotains given detailes

->input is the csv of format given in zip file

->Parsing the csv by state like shift in the previous code,we can run all slice of each shift concurrently

->Initially calculating distance of each package from ware-house

->Next declaring a qtree that came with package paulmach/go.geo/quadtree for each slice

->calling the ClusterProxitmity function

->In this we find all point that close enough of 100km,upto a limit of 50 that can be changed

-> After getting those points(kpointers) we take the largest vehicle type the we had and add the packages that are in the kpointers after clustering that pointers,we see if we could use vehicle that is smaller and assigns it that vehicle

-> this could be still developed based on needs