Advanced Programming 1 - 89-210 - Ex3

December 8, 2016

Due date: 18.12.16

1 Intro

In this exercise, you will implement your design from the previous exercises.

2 Implementation

The system will get its input through the console. Every command will be represented as a number, and its relevant parameters will be given in the next row (if there are any). The format of each command will be described later. A small change is added to this exercise - obstacles. There will be some points on the grid which will be unreachable - meaning the driver won't be able to pass through them.

You can assume for now that every given trip will start in a location with a driver in it.. If several drivers are at the same location, the first one which was created / arrived to that point, gets the ride. All drivers (and vehicles) are created at the starting point (0,0).

- Make sure all your tests from last exercise pass perfectly. You can add more if you feel you need some more.
- The coefficient of a normal can is 1, and of the luxury can is 2.

2.1 Input

The interaction with the "user" will be based on inputs. So, every turn, you will receive an input (an integer) and you should make some operation accordingly.

The first thing which the program takes as input is the size of the grid. The next thing, is the number of obstacles inside the map. Then, if there are any obstacles, the program will expect their locations. After all the relevant input regarding the grid is given, the program should expect a number which represents a command.

Here are all the possible inputs you would get (you need to support every one of them):

- 1 - insert a driver in the following format:

(id,age,status,experience,vehicle_id) - (int,int,char:{S,M,D,W},int,int)

- 2 - insert a new ride:

(id,x_start,y_start,x_end,y_end,num_passengers,tariff) - (int,int,int,int,int,int,double)

- 3 - insert a vehicle:

(id,taxi_type,manufacturer,color) - (int,{1: Normal Cab,2: Luxury Cab},char:{H,S,T,F},char:{R,B,G,P,W})

- 4 - request for a driver location:

(driver_id)
out: driver location in the format: '(x,y)'

- 6 - start driving (no input afterwards. Meaning getting all drivers to their end point)
- 7 - exit (cleaning up the program and exiting)

2.2 Output

only if needed. In this exercise, only for option number 4.

2.3 Example

input/output example:

```
3 3
0
3
0,1,H,G
1
0,30,M,1,0
2
0,0,0,0,2,1,20
6
4
0
(0,2)
7
```

Translation:
3 3 - a 3 on 3 grid
0 - with no invalid points
3 - insert a vehicle
0,1,H,G - id=0,Cab_Type=1 - Normal Cab,Manufacturer=Honda,Color=Green
1 - insert a driver
0,30,M,1,0 - id=0,age=30,status=Married,experience=1 (year),vehicle_id=0
2 - insert a trip
0,0,0,0,2,1,20 - id=0,x_start=0,y_start=0,x_end=2,y_end=2,num_passengers=1,tariff=20
6 - start driving
4 - get driver location
0 - driver with id=0
7 - exit
More examples on piazza resources (and submit system)

Submission:

There are two submissions needed.

1. All tests from last exercise (more will be accepted with pleasure). Same as last time, there will be two possible submit options - for makefile and for *cmake*. (The makefile will be compiled when you submit and you'll get a feedback, opposed to the *cmake* which will be compiled while grading. So it is your responsibility to make sure it works). This exercise it also need to run correctly, so it will obviously need also all the source code. The two possible submission (please submit only to one of them) are:

   (a) ex3_tests_make - for the makefile submisison

   (b) ex3_tests_cmake - for the cmake submission

2. Only your source code (without the tests). **This should be submitted with a makefile ONLY**. The output should be the default '*a.out*' file

As always, attach the details file to every submission.

Notes

- Don't be late.
- Submit your work via the submit system, with your personal user account. **ONLY ONE SUBMISSION PER GROUP**. A duplicated submission will be penalized. Same goes for a submission without 'details.txt' file [in every submission].
- You can already start working with Git. It is not mandatory yet, but it will get so next exercise. FYI, Git, or other Source Control tools, are being used on a daily basis. Don't be afraid of it, embrace it, cause you ain't gonna get rid of it any time soon.
- You can assume in this exercise that the input will be valid (ints for ints, chars for chars and so on...), the logical order of the input will be valid (you won't get a driver before you got it's car id) etc.
- You can assume that there will be a valid path for every trip.
- You can assume that the number of vehicles is the same as the number of drivers, and each driver gets a vehicle (until the end of the program)
- In this exercise, all drivers (and their cars) will start at the point (0,0).
- When you get the Start command (num. 6) you'll assign a trip to a driver by the order you got them. So, if the order you got the trips and drivers are: driver (0), trip (0), trip (1), trip (2), driver (1), driver 0 will get trip 0, driver 1 will get trip 1, and trip 2 won't be assigned to anyone.
- Option number 6, which indicates the beginning of the ride, will get every driver to its trip final point.

- You can use the boost external library. It's a very big library, and has numerous useful tools you can use for this exercise. If you do chose to use it, you might need to compile your files a bit differently, so make sure you add this to your makefile or cmake file. The installed version on the u2 server (and submit) of boost is 1.62.0. Make sure you install the same one (or don't use functionality of a newer version).
- Don't forget to free all memory you allocated during the program.

Grading notes:

The main measurements by which grading would be made are the following:

- C++ correct code writing.
- General working flow of the program.