مدرس: **رامتین خسروی**
طراحان: سروش میرزاسروری، ادیب رضایی، امین ستایش، علی زارع، حسام اسدالله‌زاده، علی عطاالهی

مهلت تحویل: دوشنبه ۹ خرداد، ساعت ۲۳:۵۵

Smart UTAXI!

مقدمه

در فاز اول شما به پیاده‌سازی منطق برنامه درخواست سفر و استفاده از رابط خط فرمان[1] پرداختید. در این فاز همچنان تعاملات کاربران راننده و مسافر با برنامه از طریق خط فرمان انجام خواهد شد. در این فاز امکاناتی برای محاسبه هزینه و پیشنهاد سفرها به ترتیب قیمت به راننده را پیاده‌سازی می‌کنید.

[1] CLI (Command Line Interface)

شرح تمرین

در این فاز از تمرین قابلیت محاسبه قیمت سفرها را پیاده‌سازی می‌کنید و به راننده سفرها را به همراه قیمت پیشنهاد می‌دهید تا راننده سفر را با مشاهده قیمت قبول کند. دقت داشته باشید که بعضی از دستوراتی که در فاز قبلی پیاده‌سازی کرده‌اید نیاز به تغییر دارد که در صورت تمرین توضیحات مربوط به این تغییرات ذکر شده است.

• پاسخ دستورات

به ازای هر دستوری که اقدام به اجرای آن می‌کنیم، پاسخی از سمت سیستم دریافت می‌کنیم. این پاسخ دقیقا مشابه فاز اول است و شامل پاسخ درخواست موفقیت‌امیز، پاسخ خالی بودن، پاسخ عدم وجود، پاسخ درخواست اشتباه و پاسخ عدم دسترسی می‌شود.

لیست مناطق شهر

این فایل مشابه فایل اطلاعات مناطق فاز قبلی است با این تفاوت که اطلاعاتی به آن اضافه شده است. علاوه بر نام منطقه و موقعیت جغرافیایی، در این تمرین ضریب ترافیک منطقه نیز به این فایل اضافه شده است. نمونه‌ای از این فایل در سامانه در کنار صورت پروژه قرار گرفته است. ضریب ترافیک منطقه عددی صحیح است که مشخص می‌کند درجه ترافیک منطقه‌ی موردنظر چقدر است. در ادامه پروژه با نحوه استفاده از این عدد برای محاسبه قیمت آشنا خواهید شد.

وارد کردن لیست مناطق

این بخش مانند فاز قبل است و دستور آغاز برنامه به شکل زیر خواهد بود.

توجه کنید نام فایل اجرایی برنامه شما باید utaxi.out باشد.

ورودی

```
./utaxi.out <csv_file_relative_address>
```

ورودی نمونه

```
./utaxi.out folder1/folder2/info.csv
```

محاسبه قیمت سفر

در این بخش با استفاده از اطلاعاتی که در فایل csv ورودی در اختیار شما قرار داده شده بود قیمت سفر را محاسبه می‌کنید و در دستور مربوط به محاسبه قیمت سفر آن را نمایش می‌دهید. این دستور در ادامه آورده شده است. دقت داشته باشید که این دستور صرفا قیمت محاسبه شده را نمایش می‌دهد و درخواست سفر همچنان در دستور خود که در فاز قبلی پیاده‌سازی کرده‌اید انجام می‌شود.

محاسبه قیمت سفر وابسته به ضریب ترافیک، فاصله جغرافیایی و قیمت ثابت اولیه است. در ادامه نحوه محاسبه فاصله جغرافیایی با استفاده از longitude و latitude مبدا و مقصد توضیح داده شده است.

در واقعیت برای محاسبه فاصله با استفاده از latitude و longitude از تابع‌های مثلثاتی استفاده می‌شود، اما در این تمرین برای پیچیده نشدن محاسبات فرض می‌کنیم که فاصله اقلیدسی تقریب مناسبی است. فاصله دو موقعیت به صورت زیر محاسبه می‌شود:

$$dist = 110.5 * \sqrt{(lat2 - lat1)^2 + (lng2 - lng1)^2}$$

در این تقریب (lat1, lng1) و (lat2, lng2) به ترتیب موقعیت جغرافیایی موقعیت اول و دوم هستند و 110.5 طول یک درجه در واحد کیلومتر است.

پس از محاسبه این فاصله با استفاده از رابطه‌ی زیر می‌توانید قیمت را محاسبه کنید.

$$price = dist * (traffic1 + traffic2) * 10000$$

در این رابطه traffic1 و traffic2 ضریب ترافیک موقعیت‌های مبدا و مقصد است که در فایل اولیه در اختیار قرار داده شده بود و عدد 10000 مقدار ثابت اولیه قیمت است.

دقت داشته باشید مسافران می‌توانند درخواست سفر خود را با حالت عجله دارم ثبت کنند که در این حالت باعث می‌شود قیمت در ضریب ثابت 1.2 ضرب شود و بتوانند با افزایش قیمت شانس قبول درخواست خود توسط رانندگان را افزایش دهند. این موضوع با پارامتر in_hurry که در مثال دستور نیز ذکر شده است مشخص می‌شود. همچنین دقت داشته باشید که این دستور مربوط به کاربران مسافر است و اگر نام مبدا، مقصد یا کاربر در سیستم نباشد خطای Not Found چاپ می‌شود.

ورودی
`GET cost ? username <username> origin <origin_name> destination <destination_name> in_hurry <yes/no>`
خروجی
`<calculated_price> | Bad Request | Permission Denied | Not Found`
ورودی نمونه
`GET cost ? username ali origin Amirabad destination Fatemi in_hurry yes`
خروجی نمونه
`30250.35`

درخواست سفر

این دستور تفاوتی با دستوری که در بخش قبل پیاده‌سازی کردید ندارد و صرفا باید آرگومان in_hurry را به آن اضافه کنید.

ورودی
`POST trips ? username <username> origin <origin_name> destination`

<destination_name> in_hurry <yes/no>
خروجی
<trip_id> | Bad Request | Not Found | Permission Denied
ورودی نمونه
POST trips ? username ali origin Enghelab destination Valiasr in_hurry yes
خروجی نمونه
3

لیست سفرها

این دستور نیز مانند قبل است و تغییری که در این فاز انجام می‌دهید این است که قیمت محاسبه شده را نیز برای آن نمایش دهید. همچنین باید قابلیت نمایش سفرها به ترتیب بیشترین قیمت تا کمترین قیمت وجود داشته باشد که با آرگومان sort_by_cost مشخص می‌شود.

ورودی
GET trips ? username <username> sort_by_cost <yes/no>
خروجی
<trip_id> <passenger_name> <origin> <destination> <cost> <status> ... | Empty | Permission Denied
ورودی نمونه
GET trips ? username amin sort_by_cost yes
خروجی نمونه
1 soroush Amirabad Fatemi 32564.23 waiting 2 ali Qods Vesal 10245.56 traveling

مشخصات یک سفر

در این دستور نیز قیمت سفر در این فاز اضافه می‌شود.

ورودی
`GET trips ? username <username> id <trip_id>`
خروجی
`<trip_id> <passenger_name> <origin> <destination> <cost> <status>` `... | Not Found | Permission Denied`
ورودی نمونه
`GET trips ? user amin id 2`
خروجی نمونه
`2 ali Qods Vesal 12564.00 traveling`

نحوه‌ی تحویل و نکات پایانی

- تمام فایل‌های خود را در قالب یک **پرونده‌ی زیپ** با نام A7-<SID>.zip در صفحهٔ Elearn درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۰۰۹۹۹ است، نام پروندهٔ شما باید A7-810100999.zip باشد.
  - برای مثال، نمونه فایل مورد قبول در زیر آمده است:

```
A7-810100999.zip
├── main.cpp
```

```
├── makefile
└── ...
```

- از آن جایی که در فازهای بعدی شما باید رابط کاربری برنامه‌ی خود را از command line به روش‌هایی دیگر تغییر دهید بهتر است تا طراحی برنامه‌ی شما طوری باشد که کمترین وابستگی میان منطق برنامه و رابط کاربری آن وجود داشته باشد.
- **دقت کنید** که پرونده زیپ آپلودی شما باید پس از Unzip شدن شامل پرونده‌های پروژه شما (از جمله Makefile) باشد و از زیپ کردن پوشه‌ای که داخل آن فایل‌های پروژه‌تان قرار دارد خودداری فرمایید.
- برنامهٔ شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++11 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- **دقت کنید** که پروژه شما باید Multi-file باشد و Makefile داشته باشد. همین‌طور در Makefile خود مشخص کنید که از استاندارد c++11 استفاده می‌کنید.
- درستی برنامه‌ی شما از طریق **آزمون‌های خودکار** سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- **دقت کنید** که نام پرونده‌ی اجرایی شما باید **utaxi.out** باشد.
- طراحی درست، رعایت سبک برنامه نویسی درست و تمیز بودن کد برنامه‌ی شما در نمره‌ی تمرین تأثیر زیادی دارد.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.
- سوالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن‌ها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این فاز پروژه ارتباط برقرار کنید.

- دقت کنید که تمام اعداد غیر صحیح که در خروجی چاپ می‌شوند باید **دقت 2 رقم اعشار بدون گردکردن** داشته باشند. (مثلا عدد 2.1 باید 2.10 و عدد 4.1234 باید 4.12 نمایش داده شود.)
- **توجه داشته باشید که حالت‌های خاصی که در صورت پروژه ذکر نشده است در تست‌های خودکار نخواهد بود و می‌توانید به هر شکلی که مد نظر دارید آن‌ها را مدیریت کنید.**