برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۶

مدرس: **رامتین خسروی**

طراحان: مهدی بهلول، محمد فرهی، علی پادیاو، امیرعلی رحیمی، آوا میرمحمدمهدی، سارا رضائی‌منش، شهنام فیضیان

مهلت تحویل: سه‌شنبه ۲۶ اردیبهشت ۱۴۰۲، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین آشنایی شما با مفاهیم اولیه **چند ریختی** و طراحی یک نرم‌افزار به کمک آن است. طراحی کلاس‌ها، نحوهٔ ارث‌بری آن‌ها از یکدیگر و تعریف صحیح توابع مربوط به هر کدام از کلاس‌ها اهمیت بالایی دارد؛ به همین منظور پیشنهاد می‌شود قبل از پیاده‌سازی پروژه، ابتدا طراحی‌های مختلف را بررسی و سپس مناسب‌ترین طراحی را پیاده‌سازی کنید.

سامانه ماموریت راننده

در این سامانه قصد داریم تا برای افزایش فعالیت رانندگان یک سامانه آنلاین درخواست خودرو (مانند اسنپ)، سیستم ماموریت‌دهی به راننده‌ها را طراحی کنیم. این سیستم به این نحو عمل می‌کند که تعدادی ماموریت از پیش تعریف شده داریم و هر راننده می‌تواند یک الی چند ماموریت داشته باشد. راننده‌ها با انجام سفر می‌توانند ماموریت خود را تکمیل کنند و پاداشی دریافت کنند.

سفر

هر سفر مسیری است که راننده از مبدایی خاص تا مقصدی خاص را طی می‌کند و شامل ویژگی‌های زیر است:

- برچسب زمانی شروع سفر
- برچسب زمانی پایان سفر
- شناسه راننده
- مسیر طی شده در سفر به متر

ماموریت

در سامانه ماموریت‌ها با ویژگی‌های مختلفی وجود دارد؛ برای مثال ماموریتی وجود دارد که شرط پایان یافتن آن، این است که راننده باید در بازه‌ی زمانی مشخص، حداقل n تعداد سفر کرده باشد؛ یا راننده باید در بازه‌ی زمانی مشخص، حداقل n متر در سفرها پیموده باشد. همچنین هر راننده با اتمام ماموریت خود، به پاداشی که در ماموریت تعیین شده است، می‌رسد.

هر ماموریت دارای چند ویژگی اصلی می‌باشد:

- شناسه ماموریت
- برچسب زمانی شروع ماموریت
- برچسب زمانی پایان ماموریت

ماموریت زمانی

در این مدل ماموریت، راننده باید در بازه‌ی زمانی مشخص ماموریت حداقل n دقیقه را در طول سفرهای خود گذرانده باشد.

ماموریت مسافتی

در این مدل ماموریت، راننده باید در بازه‌ی زمانی مشخص ماموریت حداقل n متر را در طول سفرهای خود گذرانده باشد.

ماموریت تعدادی

در این مدل ماموریت، راننده باید در بازه‌ی زمانی مشخص ماموریت حداقل n تعداد سفر را به پایان رسانده باشد.

دستورات

- **ایجاد ماموریت زمانی جدید**

قالب ورودی

```
add_time_mission <mission_id> <start_timestamp> <end_timestamp> <target_time_in_minutes> <reward_amount>
```

قالب خروجی

```
OK
```

- در صورت تکراری بودن پارامتر mission_id، باید خطای DUPLICATE_MISSION_ID نمایش داده شود.
- در صورتی که زمان پایان ماموریت کوچک‌تر از زمان شروع آن باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.

- در صورتی که تعداد دقیقه‌ها یا مقدار پاداش عددی کوچکتر از صفر باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.

نمونه ورودی

```
add_time_mission 1 1680347116 1980349516 10000 200000
```

نمونه خروجی

```
OK
```

- **ایجاد ماموریت مسافتی جدید**

قالب ورودی

```
add_distance_mission <mission_id> <start_timestamp> <end_timestamp> <target_distance_in_meters>
<reward_amount>
```

قالب خروجی

```
OK
```

- در صورت تکراری بودن پارامتر mission_id، باید خطای DUPLICATE_MISSION_ID نمایش داده شود.
- در صورتی که زمان پایان ماموریت کوچک‌تر از زمان شروع آن باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.
- در صورتی که مسافت تعیین شده یا مقدار پاداش عددی کوچکتر از صفر باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.

نمونه ورودی

```
add_distance_mission 3 1076347116 1580349516 250000 200000
```

نمونه خروجی

```
OK
```

- **ایجاد ماموریت تعدادی جدید**

قالب ورودی

```
add_count_mission <mission_id> <start_timestamp> <end_timestamp> <target_number> <reward_amount>
```

قالب خروجی

```
OK
```

- در صورت تکراری بودن پارامتر mission_id، باید خطای DUPLICATE_MISSION_ID نمایش داده شود.
- در صورتی که زمان پایان ماموریت کوچک‌تر از زمان شروع آن باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.
- در صورتی که تعداد تعیین شده سفرها یا مقداری پاداش عددی کوچکتر از صفر باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.

نمونه ورودی

```
add_count_mission 2 1076347116 1580349516 70 200000
```

نمونه خروجی

```
OK
```

- **اختصاص ماموریت به راننده**

با این دستور، می‌توان یک ماموریت را به راننده اختصاص داد.

قالب ورودی

```
assign_mission <mission_id> <driver_id>
```

قالب خروجی

```
OK
```

نمونه ورودی

```
assign_mission 1 1
```

نمونه خروجی

```
OK
```

- اگر شناسه ماموریت وارد شده متناظر با هیچ ماموریتی در سیستم نباشد، باید پیغام MISSION_NOT_FOUND چاپ شود.
- اگر راننده با شناسه متناظر در سیستم موجود نباشد، یک راننده جدید با شناسه مورد نظر در سیستم در نظر گرفته می‌شود.

- اگر ماموریتی با این شناسه در حال حاضر به راننده اختصاص داده شده بود، خطایی مبنی بر DUPLICATE_DRIVER_MISSION برگردانده می‌شود.

- **اطلاع سیستم از سفر به اتمام رسیده**

با این دستور، می‌توان اطلاعات یک سفر پایان یافته را در سیستم وارد کرد و لیست ماموریت‌هایی که با پایان این سفر به پایان رسیده‌اند را به عنوان خروجی بازگرداند.

قالب ورودی

```
record_ride <start_timestamp> <end_timestamp> <driver_id> <distance>
```

قالب خروجی

```
completed missions for driver <driver-id>:
mission: <mission-id>
start timestamp: <start-timestamp>
end timestamp: <end-timestamp>
reward: <reward-amount>

mission: <mission-id>
…
```

- فیلد end-timestamp برابر با زمانی است که آن ماموریت برای یک راننده‌ی خاص به پایان می‌رسد. در اینجا برای راحتی کار، برابر با زمان پایان سفری می‌باشد که تمام کننده‌ی ماموریت بوده است.

نمونه ورودی ۱

```
record_ride 1680347116 1680349516 1 20000
```

نمونه خروجی ۱

```
completed missions for driver 1:
mission: 1
start timestamp: 1670349516
end timestamp: 1680349516
reward: 200000
```

نمونه ورودی ۲

```
record_ride 1680347116  1680349516 2 2000
```

نمونه خروجی ۲

```
completed missions for driver 2:
```

- نمونه خروجی ۲ نشانگر حالتی می‌باشد که با اتمام سفر، هیچ ماموریتی به پایان نمی‌رسد.
- دستور به همراه همه آرگومان‌هایش در یک خط به برنامه داده می‌شود.
- تمام آرگومان‌ها به صورت عدد صحیح می‌باشند.
- اگر راننده‌ی مشخص شده در اطلاعات سفر دارای **هیچ** ماموریتی نبود، اطلاعات این سفر در نظر گرفته نخواهد شد و تاثیری در برنامه نخواهد داشت.
- در صورتی که زمان پایان سفر کوچک‌تر از زمان شروع آن باشد، باید خطای INVALID_ARGUMENTS نمایش داده شود.
- ماموریت‌ها به ترتیب برچسب زمانی شروع آنها نمایش داده می‌شوند.

- **گزارش وضعیت ماموریت‌های یک راننده**

با این دستور، لیست ماموریت‌های یک راننده به همراه مشخصات هر ماموریت به ترتیب برچسب زمانی شروع آنها نمایش داده می‌شوند.

قالب ورودی

```
show_missions_status <driver-id>
```

قالب خروجی

```
missions status for driver <driver-id>:
mission <mission-id>:
start timestamp: <start-timestamp>
end timestamp: <end-timestamp>
reward: <reward-amount>
status: <completed|ongoing>

mission <mission-id>:
...
```

- اگر راننده‌ی مشخص شده در دستور دارای هیچ ماموریتی نباشد، باید خطایی مبنی بر DRIVER_MISSION_NOT_FOUND نمایش داده شود.

نمونه ورودی

```
show_missions_status 1
```

نمونه خروجی

```
mission status for driver 1:
mission 1:
start timestamp: 1680347116
reward: 20000
status: completed
```

نکات تکمیلی

- دقت داشته باشید که تمامی برچسب‌های زمانی در برنامه با دقت ثانیه خواهند بود.

- در تمامی دستورات، اگر مقادیری که از نوع رشته‌ای از کاراکترها هستند، دارای هیچ کاراکتری نباشند یا null باشند، باید مقدار INVALID_ARGUMENTS نمایش داده شود.
- طراحی درست وراثت، رعایت سبک برنامه‌نویسی درست و تمیز بودن کد برنامه‌ی شما در نمره‌ی تمرین تأثیر زیادی دارد. برای مثال استفاده از if یا switch case برای تشخیص نوع زیرکلاس یک کلاس پدر نشان‌دهنده طراحی نادرست وراثت است.
- در تمامی مراحل این پروژه سعی کنید از قوانین ارث‌بری استفاده کنید و هر جا که ممکن است رفتار کلاس‌ها را به صورت چندریخت (polymorphic) پیاده‌سازی کنید و از بررسی مجزای کلاس‌ها خودداری کنید.

نحوهٔ تحویل

- به غیر از خطاهای ذکر شده در صورت پروژه، نیاز به رسیدگی به هیچ خطای دیگر نمی‌باشد و تضمین می‌شود ورودی‌ها به درستی به برنامه داده می‌شوند.
- پرونده‌های خود را در قالب یک پرونده‌ی zip با نام A6-<SID>.zip در صفحهٔ elearn درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۰۱۹۹۹ است، نام پروندهٔ شما باید A6-810101999.zip باشد.
- از zip کردن پوشه‌ای که داخل آن فایل‌های پروژه‌تان قرار دارد خودداری فرمایید.
- دقت کنید که پروژه‌ی شما باید Multi-file باشد و Makefile داشته باشد. همین‌طور در Makefile خود مشخص کنید که از استاندارد C++11 استفاده می‌کنید.
- دقت کنید که نام پرونده‌ی اجرایی شما باید main باشد.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.