برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۴

مدرس: رامتین خسروی

طراحان: علی حمزه‌پور، حسام رمضانیان، سروش صحرائی، علی عطاءاللهی، شهنام فیضیان، امیررضا نادی، محمد امین یوسفی

مهلت تحویل: سه‌شنبه ۴ اردیبهشت ۱۴۰۳، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین آشنایی شما با مفاهیم اولیه **طراحی شیءگرا**[1] و طراحی یک نرم‌افزار به کمک آن است. از آنجایی که استفاده و درک درست این مفاهیم در پیاده‌سازی سایر تمرین‌های این درس لازم است، پیشنهاد می‌شود به این تمرین زمان کافی را اختصاص دهید.

سامانه مدیریت کارواش

شرح مسئله

در این پروژه قرار است سامانه‌ای برای مدیریت یک کارواش طراحی کنید تا وضعیت کارکنان، ماشین‌ها و بخش‌های مختلف کارواش را نگه دارد و به درستی به‌روز کند. ماشین‌ها برای طی کردن چندین مرحله شست‌وشو وارد کارواش می‌شوند. هر ماشین به ترتیب وارد مراحل مختلف جهت شست‌وشو می‌شود و در صورتی که در هر مرحله کارگر مناسب برای انجام کار ماشین وجود نداشته باشد، ماشین وارد صف آن مرحله می‌شود تا زمانی که یک کارگر به آن اختصاص یابد. در نهایت پس از طی آخرین مرحله، ماشین از کارواش خارج می‌گردد.

زمان

در سامانه ما زمان نقش مهمی دارد و باید به درستی شبیه‌سازی شود. برای راحتی شما ساده‌سازی‌هایی درباره زمان فرض شده که در ادامه به آن‌ها اشاره خواهد شد. واحد زمانی در این پروژه به صورت خطی در نظر گرفته شده و زمان **یک واحد یک واحد** جلو می‌رود. بنابراین مبدا زمانِ سامانه عدد ۰ خواهد بود.

[1] Object-Oriented Design

کارگرها

کارگرهای کارواش همواره در کارواش حضور دارند و می‌توانند روی ماشین‌ها کار کنند. هر کارگر در یک بخش خاص کار می‌کند و نمی‌تواند روی ماشینی که در بخش دیگری قرار دارد کار کند. یکی از وظایف سامانه ما اختصاص دادن کارگر مناسب به ماشین‌ها است. توجه داشته باشید که اگر برای انتصاب کارگر، چند کارگر بیکار داشتیم، اولویت با کارگری است که زمان کم‌تری طول می‌کشد تا کارش را تمام کند و در صورت برابر بودن زمان اتمام کار، کارگری که شناسه یکتای کوچکتری دارد اولویت دارد.

مراحل شست‌وشو

کارواش چندین مرحله شست‌وشو دارد که ماشین‌ها در آن‌ها توسط کارگران شسته می‌شوند. هر مرحله یک قیمت دارد که مشتری باید به ازای آن مرحله بپردازد. تعداد ماشین‌هایی که هم‌زمان می‌توانند در یک مرحله باشند، وابسته به تعداد کارگران فعال آن مرحله است. به عبارتی اگر کارگری در آن مرحله بیکار باشد، ماشین می‌تواند وارد آن مرحله شود. در صورت نبود کارگر مناسب برای انتصاب، ماشین وارد صف آن مرحله می‌شود و منتظر می‌ماند تا کارگر مناسب برای آن پیدا شود. اگر چندین ماشین هم‌زمان بخواهند وارد مرحله یا صف مرحله شوند، اولویت با ماشینی است که شناسه یکتای کمتری دارد.

مشتریان

مشتریان در یک زمان مشخص به کارواش مراجعه می‌کنند و می‌خواهند که در چندین مرحله شست‌وشو حضور پیدا کنند. ترتیب حضور پیدا کردن آنها در این مراحل شست‌وشو با توجه به ترتیبی که خودشان مشخص کرده‌اند انجام خواهد شد. سامانه ما باید به ماشین‌ها با توجه به مرحله شست‌وشو فعلی کارگر اختصاص دهد و یا آن‌ها را در صف انتظار مرحله قرار دهد. توجه داشته باشید که انتقال ماشین‌ها از یک مرحله به مرحله بعد و یا انتقال آن‌ها از صف انتظار به خود مرحله و اختصاص دادن کارگر مناسب به آنها را نیز مدل کنید.

فایل‌های ورودی

بخشی از اطلاعات مورد نیاز برنامه در قالب فایل‌های csv به برنامه داده می‌شود که باید اطلاعات موجود در آن‌ها را در ابتدای اجرا برنامه خوانده و ذخیره کنید. در ادامه شرح هر یک از فایل‌ها آمده است.

مراحل شست و شو

نوع داده	توضیحات	نام ویژگی
عدد طبیعی	شناسه یکتا مرحله	Id
عدد طبیعی	قیمت مرحله	Price

کارگرها

نوع داده	توضیحات	نام ویژگی
عدد طبیعی	شناسه یکتا کارگر	Id
عدد طبیعی	شناسه مرحله‌ای که کارگر در آن کار می‌کند	Stage-id
عدد طبیعی	زمان مورد نیاز برای انجام هر شستشو توسط کارگر	Time-to-finish

خط اول فایل‌های csv مختص به header آن فایل خواهد بود و تضمین می‌شود که ترتیب فیلدها مشابه با ترتیب ذکر شده در صورت پروژه است. مسیر این فایل‌ها توسط آرگومان‌های خط فرمان به برنامه داده می‌شود. برای آشنایی با آرگومان‌های خط فرمان به این لینک مراجعه کنید. مسیر مربوط به فایل‌ها با قالب زیر به برنامه داده می‌شود.

```
./a.out </path/to/stages/file> </path/to/workers/file>
```

مثال :

```
./a.out Stages.csv Workers.csv
```

دستورات ورودی و قالب خروجی

جلو بردن زمان

در این دستور زمان به اندازه گفته شده به جلو پیش می‌رود. شما باید گزارشی از تغییر وضعیت ماشین‌ها که در این مدت اتفاق افتاده را به کاربر بدهید. هر خط از گزارش شامل زمان اتفاق، ماشینی که وضعیت آن تغییر کرده و شرح اتفاق است.
گزارش‌ها بر اساس زمان اتفاق (از زودتر به دیرتر) چاپ می‌شوند و در صورت هم‌زمان بودن دو اتفاق اولویت گزارش‌دهی با آنی است که شناسه یکتای کوچکتری دارد. توجه داشته باشید که هیچ حالتی نداریم که در یک زمان دو اتفاق برای یک ماشین خاص گزارش شود.
تمامی حالات ممکن برای گزارش‌دهی در بخش نمونه خروجی آورده شده، لطفا با دقت به آن توجه کنید.

قالب دستور

```
pass_time <number_of_time_units_to_pass>
```

قالب خروجی

```
<time> Car <ID>: <pervious_position> -> <current_position>
```

نمونه ورودی

```
pass_time 3
```

نمونه خروجی

```
18 Car 2: Stage 3 -> Done
19 Car 3: Stage 2 -> Stage 4
19 Car 4: Stage 1 -> Queue 5
20 Car 1: Queue 1 -> Stage 1
20 Car 4: Queue 5 -> Stage 5
```

ورود ماشین

این دستور یک ماشین با مراحل شست‌وشو مشخص را در زمان فعلی به کارواش اضافه می‌کند. تعداد مراحل شست‌وشو عدد ثابتی نیست و می‌تواند برای ماشین‌های مختلف متفاوت باشد. این مراحل شست‌وشو به عنوان آرگومان‌های دستور می‌آیند و با فاصله[2] از هم جدا می‌شوند. تضمین می‌شود مراحل شست‌وشو درخواست شده توسط ماشین حتما وجود دارد. شما باید یک سیستم ساده تخصیص شناسه یکتا به ماشین‌ها نیز پیاده‌سازی کنید، به این صورت که ماشین اول شناسه ۱ را دریافت می‌کند و شناسه ماشین‌های بعدی به ترتیب یکی بیشتر از ماشین قبلی خواهند بود.

قالب دستور

```
car_arrival <stagesIDs>
```

قالب خروجی

```
<time> Car <carID>: Arrived -> <position>
```

نمونه ورودی

```
car_arrival 1 4 6 7
```

نمونه خروجی

```
4 Car 12: Arrived -> Stage 1
```

```
4 Car 12: Arrived -> Queue 1
```

بررسی وضعیت یک مرحله

در این دستور درباره وضعیت یک مرحله اعم از تعداد ماشین‌های شسته شده، صف ماشین‌های منتظر برای شست‌وشو، ماشین‌هایی که الان در حال شست‌وشو هستند و درآمد مرحله (دقت کنید هر ماشین بعد از رد شدن از یک مرحله، به اندازه قیمت آن مرحله، به آن مرحله پول پرداخت می‌کند) از شما پرسیده می‌شود.

---

[2] Space

قالب دستور

```
get_stage_status <stageID>
```

قالب خروجی

```
Number of washed cars: <Count>
Number of cars in queue: <Count>
Number of cars being washed: <Count>
Income: <Amount>
```

نمونه ورودی

```
get_stage_status 4
```

نمونه خروجی

```
Number of washed cars: 1
Number of cars in queue: 3
Number of cars being washed: 4
Income: 234
```

بررسی وضعیت یک کارگر

این دستور از شما درباره وضعیت یک کارگر می‌پرسد. اگر آماده به کار باشد عبارت (Idle) و در صورتی که درحال کار روی یک ماشین بود عبارت (Working: <carID>) چاپ می‌شود.

قالب دستور

```
get_worker_status <workerID>
```

قالب خروجی

```
Idle
```

```
Working: <carID>
```

نمونه ورودی

```
get_worker_status 9
```

نمونه خروجی

```
Working: 3
```

بررسی وضعیت یک ماشین

با این دستور وضعیت یک ماشین را بررسی می‌کنیم. وضعیت ماشین سه حالت دارد:

- In line: در صف یک مرحله است
- In service: در حال طی کردن یک مرحله است
- Done: کارش تمام شده

در مرحله In line و In service نام مرحله مربوطه نیز ذکر شود.

قالب دستور

```
get_car_status <carID>
```

قالب خروجی

```
<In line/In service/Done>[: Stage/QueueID]
```

نمونه ورودی

```
get_car_status 5
```

نمونه خروجی

```
In line: 3
```

نکات

- در صورت عدم وجود ماشین، مرحله یا کارگر در همه دستورات عبارت NOT FOUND چاپ شود.
- درآمد مراحل هنگامی که کار ماشین در آن مرحله تمام شد اضافه می‌شود.
- زمان اتمام کار قبلی کارگر می‌تواند با زمان شروع کار جدیدش یکی باشد. (مثال: کارگر در زمان ۲۰ کارش روی ماشین ۴ تمام می‌شود و در همان زمان ۲۰ کارش را روی ماشین ۷ شروع می‌کند)
- خطایی جز خطاهایی که در صورت پروژه به آن‌ها اشاره شده، در تست کیس‌ها وجود نخواهد داشت و نیازی به مدیریت کردن خطاهای دیگر نیست.

نکات و نحوهٔ تحویل

- فایل‌های خود را در قالب یک فایل با نام A4-SID.zip در صفحهٔ eLearn درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۰۲۰۰۰ باشد، نام فایل شما باید A4-810102000.zip باشد که شامل کدهای پروژه شما است.
- پروژه شما باید به صورت **چند فایلی**[3] و با استفاده از **makefile** پیاده‌سازی شده باشد. هدف اصلی پروژه یادگیری شی‌گرایی بوده و پیاده‌سازی به صورت چند فایلی صرفا برای آشنایی شما با این مفهوم می‌باشد. دقت کنید در پروژه‌های بزرگ‌تر، شما از ابتدا باید فایل‌ها و اجزای مختلف پروژه را تشخیص داده و آن را پیاده‌سازی کنید؛ با این حال برای سادگی بیش‌تر و به منظور کسب تجربه، بهتر است این پروژه را ابتدا به صورت یک فایل پیاده‌سازی کرده و تمرکز خود را روی طراحی شی‌گرا بگذارید؛ پس از تشخیص کلاس‌ها و پیاده‌سازی پروژه، آن را به چند فایل تقسیم کرده و makefile مناسب را بنویسید.
- ۱۰ درصد از نمره نهایی این پروژه به پیاده‌سازی صحیح به صورت چند فایلی اختصاص داده شده است. شما **همچنان می‌توانید** پروژه خود را به صورت تک فایلی و با کسر این ۱۰ نمره تحویل دهید. در این صورت کد شما باید **یک فایل** با نام A4-SID.cpp باشد که SID شماره دانشجویی شماست.
- در makefile خود مشخص کنید که از استاندارد c++20 استفاده می‌کنید.
- نام برنامه قابل اجرای شما باید CarWashManager باشد.
- درستی برنامهٔ شما از طریق آزمون‌های خودکار سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

[3] multi-file

نمرات

- تمیزی کد
  - رعایت کردن نام‌گذاری صحیح و انسجام
  - عدم وجود کد تکراری
  - رعایت دندانه‌گذاری[4]
  - عدم استفاده از متغیرهای گلوبال
  - استفاده **صحیح** از متغیرهای ثابت به جای Magic Value-ها
- درستی کد
  - آزمون‌های خودکار
- طراحی
  - شکستن به کلاس‌های مناسب و تخصیص مسئولیت‌های صحیح به هر کلاس
  - جداسازی منطق کد از ورودی/خروجی
  - رعایت سطح دسترسی (public/private) در ویژگی‌های کلاس
  - عدم وجود منطق در تابع main
  - ساختاردهی کد در قالب توابع کوتاه که فقط یک کار را انجام می‌دهند

دقت کنید که موارد ذکر شده لزوما کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.

[4] Indentation