به نام خداوند رنگین کمان

درس سیستم های عامل

تمرین سوم عملی

مدرس : دکتر انتظاری

طراح : محمدحسین کریمیان

قوانین

دانشجویان محترم لطفاً نکات تکمیلی زیر را در تمامی تمرینها در نظر بگیرید .

•در مجموع تمام تمرینها، ۲۴۰ ساعت تأخیر در ارسال پاسخها مجاز است به همین جهت زمان اعالم شده به هیچ وجه قابل تمدید نیست و در صورت نیاز میتوانید از این زمان استفاده کنید، با پایان یافتن این زمان نمره شما ۰ لحاظ خواهد شد .

•پروژه های درسی مهلت تاخیر ندارند .

•انجام تمرین و کوییز به صورت یکنفره میباشد.

• فرمت ارسالی تمرین تئوری حتماً باید بهصورت latex یا word باشد.

• در صورت مشورت کردن در مورد سؤال خاصی ذکر اسامی همراه با سؤال موردنظر، در گزارش کار طرفین الزامی هست غیر این صورت مشاهده هرگونه تشابه در گزارش کار یا کدهای پیادهسازی، به منزله تقلب برای طرفین در نظر گرفته میشود.

• تمامی منابع استفاده شده بهصورت دقیق ذکر شوند . همچنین استفاده از کدهای آماده بدون ذکر منبع و بدون تغییر به منزله تقلب خواهد بود و نمره تمرین شما صفر در نظر گرفته میشود .

•بخش زیادی از نمره شما مربوط به گزارش کار و روند حل مسئله است. لطفاً به موارد خواسته شده به صورت کامل، دقیق و بدون ابهام پاسخ دهید .

•لطفاً گزارش، فایل کدها و سایر ضمائم موردنیاز را با فرمت زیر در سامانه مدیریت دروس بارگذاری نماید به جای n شماره تمرین قرار داده میشود HWn_[studentNumber].zip .

•درصد تشابه باالی ۳۰ درصد از دید quera ،تقلب لحاظ میشود و بعد از تکرار برای بار دوم، نمره این تمرین و تمرین قبلی که تقلب آن گرفته نشده، ۰ لحاظ خواهد شد .

•زمان ارسال کوییزهای کلاس حل تمرین تا ساعت ۱۲ همان شب خواهد بود و این مورد به هیچ وجه تمدید نخواهد شد.

•فرض کوییزهای کلاس حل تمرین این است که این امتحان بهصورت book open برگزار نمیشود و اجازه استفاده از کدهای آماده در سایت ها را نخواهید داشت.

1..در این تمرین قصد داریم با مفاهیم Thread وSynchronization بیشتر آشنا شویم.
در این سوال باید دنباله π را با استفاده از Thread پیاده سازی کنید. برنامه ای بنویسید که با استفاده از فرمول زیر دنباله π را تا تعداد معینی عبارت محاسبه کند.

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4}.$$

توجه داشته باشید که برنامه شما باید بتواند با تعدادی رشته مختلف کار کند.
به عنوان مثال، اگر شما تا 10 میلیونیم این دنباله را در برنامه خود با استفاده از 10 رشته محاسبه کنید، هر رشته باید یک میلیون عبارت از این دنباله را محاسبه کند.

2. در برنامه های دارای چند thread، شرایطی پیش می آید که در آن نیاز است ترد ها منتظر بمانند تا کار دیگران نیز به مرحله ای مشخص برسد. به عنوان مثال یک برنامه را در نظر بگیرید که میخواهد ماتریسی را به توان پنج برساند. در این برنامه هر ریسمان مسئول محاسبه یک درایه است. برای سادگی فرض کنید یک ماتریس 2 × 2 را میخواهیم به توان پنج برسانیم. هنگامی که درایه a11 برای ماتریس A^2 توسط یکی از ریسمان ها محاسبه میشود، قبل از ادامه دادن برای محاسبه توان A^3 باید صبر کند تا دیگر ریسمان ها نیز مقادیر دیگر درایه ها را محاسبه کنند. توجه کنید که عملیات join در اینجا مطلوب نیست چرا که join صبر می کند تا کار یک ریسمان به صورت کامل تمام شود. در این مواقع از یک عامل Synchronization استفاده می شود با نام مانع که به این صورت عمل می کند که بعد از ایجاد و در هنگام مقدار دهی اولیه، مشخص می شود که چند تا ریسمان باید به این مانع برسند تا اجازه عبور برای همه داده شود. به همین دلیل ریسمان هایی که زودتر به مانع می رسند متوقف می شوند تا زمانی که تعداد ریسمان های منتظر به شماره مورد نظر برسد .سپس تمام ریسمان ها اجازه عبور از مانع را خواهند داشت. کد زیر سعی می کند نحوه استفاده از یک مانع را در یک زبان فرضی نشان دهد.

```
// define a barrier
barrier_t b

function main:
    init_barrier(&b, 10) // wait until 10 threads reach the block;
    for i to 10 do
        create_thread(worker_func)
    end
    ...

function worker_func():
    process()
    ...
    wait(b) // wait until all 10 threads reach this point
    // continue processing
    ...
```

1) از شما خواسته شده است تا فقط با استفاده از mutex این عامل هماهنگ سازی را پیاده کنید. استفاده از Semaphoreو یا Conditional Variables برای این منظور مجاز نیست. در صورت امکان کد پیاده سازی را بنویسید و آن را تست کنید. درصورتی که امکان پیاده سازی این مانع با شرایط گفته شده وجود ندارد دلیل خود را به صورت مفصل و به همراه ذکر مثال شرح دهید.
2) با استفاده از semaphore یا conditional variables پیاده سازی را انجام دهید.

3. پاسخ مسئله dining-philosophers را پیاده سازی کنید و با سناریو دلخواه، درستی آن را امتحان کنید.

4. الگوریتمی طراحی کنید که در آن یک ساعت آلارم دار وجود دارد که به یک برنامه تماس این امکان را می دهد تا خود را برای تعداد مشخصی از واحدهای زمانی (tick) به تاخیر بیندازد. برای پیاده سازی میتوانید یک ساعت سخت افزاری واقعی را فرض کنید که در فواصل زمانی معین، یک تابع به نام tick() را در مانیتور شما فراخوانی می کند.