دانشکده مهندسی کامپیوتر

دکتر رضا انتظاری ملکی

پاییز ۱۴۰۰

تمرین تئوری سری چهارم

سیستم‌های عامل

غزل زمانی‌نژاد - محمدعلی فراهت

تاریخ تحویل: شنبه ۱۳ آذر ساعت ۲۳:۵۹:۵۹

قوانین

- درصورت مشاهده‌ی هرگونه تقلب، به ازای هربار تقلب نمره‌ی کل آن تمرین صفر درنظر گرفته می‌شود و همچنین یک نمره (نمره منفی) از نمره‌ی کل تمرین‌ها کسر می‌شود.
- درصورت وجود هرگونه سوال از طریق گروه تلگرام مطرح کنید. (لطفا پی‌وی پیام ندهید.)
- ۱۰ درصد از نمره‌ی هر تمرین به تمیزی و نظم پاسخ‌های ارسالی شما تعلق گرفته است. لازم است به موارد زیر توجه کنید:

۱. خوانا و مرتب بنویسید.

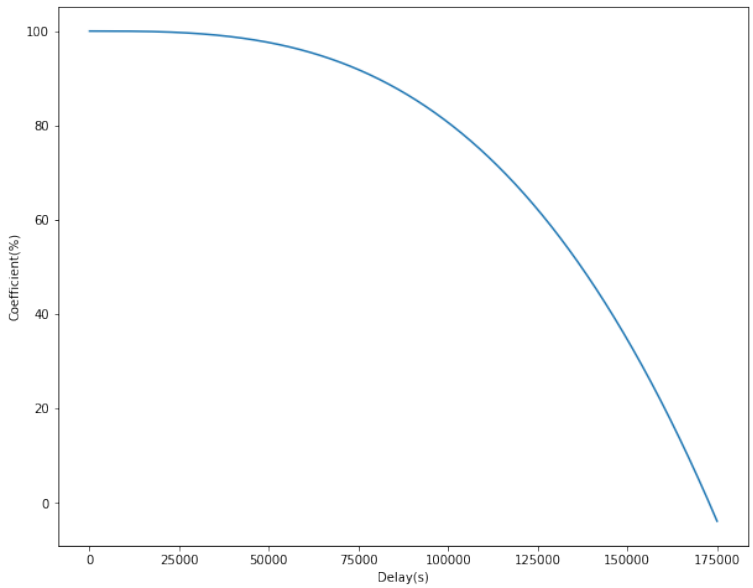
۲. از نرم‌افزارهای مناسب جهت اسکن کردن تمرین‌های خود استفاده کنید و چک کنید که نور تصاویر مناسب هستند. مانند:

CamScanner, Microsoft Office Lens, Adobe Scan, ...

۳. به طور عمودی عکاسی کنید.

۴. پاسخ هر سوال را به طور جداگانه در کوئرا آپلود کنید.

- محور افقی این نمودار، مقدار تاخیر به ثانیه و محور عمودی ضریب اعمالی در نمره تمرین است.



شکل ۱: نمودار تاخیر

سوالات

۱ Busy waiting (۱۵ نمره)

معنی busy waiting چیست؟ چه مدل‌های دیگری از waiting در سیستم عامل وجود دارد؟ آیا میتوان به طور کلی از busy waiting اجتناب کرد؟ پاسخ خود را توضیح دهید.

۲ Semaphore (۲۰ نمره)

نشان دهید که چگونه می‌توان از یک سمافور باینری برای پیاده‌سازی mutual exclusion میان n پردازه استفاده کرد.

۳ Signal operation (۱۵ نمره)

چگونه عملیات signal() مرتبط با monitors با عملیات مربوطه تعریف شده برای semaphores متفاوت است؟

۴ Monitor (۲۰ نمره)

قرار است یک فایل بین چند process به اشتراک گذاشته شود که هر کدام دارای یک شماره منحصر به فرد هستند. می‌توان به صورت همزمان توسط چندین process ، مشروط به محدودیت زیر به فایل دسترسی داشت:
مجموع اعداد منحصر به فرد هر process که در حال حاضر به فایل دسترسی دارد، کمتر از n باشد.
یک monitor برای هماهنگی دسترسی به این فایل بنویسید.

۵ Producer-Consumer problem (۲۰ نمره)

یک مسئله تولیدکننده – مصرف‌کننده با بافر محدود را در نظر بگیرید که در آن چندین تولیدکننده به صورت همزمان وجود دارند و اندازه آیتم‌های تولیدی و مصرفی با هم متفاوت است. ظرفیت کل بافر N است و این پردازه‌ها از دو سمافور شمارشی برای دسترسی به بافر استفاده می‌کنند: full دارای مقدار اولیه صفر و empty دارای مقدار اولیه N است.

هر تولیدکننده کد زیر را برای اضافه کردن آیتمی با اندازه k به بافر استفاده می‌کند:

```
for i from 1 to k do {
    P(empty)
}
//insert item of size k into buffer
V(full)
```

مصرف‌کننده نیز کد زیر را برای حذف آیتم از بافر اجرا می‌کند:

```
P(full)
//remove item of size k from the buffer
for i from 1 to k do {
    V(empty)
}
```

آیا با استفاده از این کد می‌توان اطمینان حاصل کرد که:

(آ) تولیدکننده تا زمانی که در بافر ظرفیت کافی وجود نداشته باشد آیتمی را اضافه نکند؟

(ب) مصرف‌کننده تنها زمانی که آیتمی در بافر باشد اقدام به حذف کند؟

(ج) تنها یک پردازه (تولیدکننده یا مصرف‌کننده) از بافر در یک زمان استفاده کند؟

(د) بن‌بست ایجاد نمی‌شود؟

موفق باشید.