دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

به نام خدا

تمرین سری ششم درس سیستم های عامل

پاییز 1403

استاد درس: دکتر زرندی

توضیحات:

- پاسخ به تمرین ها باید به صورت انفرادی صورت پذیرد. درصورت مشاهده هر گونه تقلب نمره صفربرای کل تمرین منظور خواهد شد.
- تمیزی و خوانایی جواب تمرین ها از اهمیت بالایی برخوردار است. لطفا این مورد را رعایت کنید تا نمره ای به این سبب از شما کسر نگردد.
- لطفا پاسخ تمرین ها را در قالب یک فایلPDF با نام "HW6_StudentNumber.pdf" در سامانه کورسز و در مهلت معین شده بارگذاری فرمایید.
- در صورت برخوردن به هرگونه مشکل در رابطه با تمرین میتواند از طریق ایمیل os.fall1403@gmail.com و یا تلگرام با تدریسیاران در ارتباط باشید.

سوال اول)

ناحیه بحرانی را تعریف کنید و شروط لازم و کافی را برای آن نام ببرید و به صورت مختصر توضیح دهید.

سوال دوم)

دو روش برای مدیریت نواحی بحرانی به صورت Preemptive و Non preemptive می‌باشد، این دو روش را توضیح دهید و برای هرکدام یک مثال بیاورید که در چه نوع سیستم‌هایی بهتر است استفاده شوند.

سوال سوم)

در رابطه با نواحی بحرانی به سوالات زیر پاسخ دهید.

الف) دستورات Atomic به چه دستوراتی گفته می شود؟

ب) دو مورد از برتری‌های استفاده از Semaphore به‌جای Mutex را توضیح دهید.

ج) الگوریتم پترسون را برای پشتیبانی از N پردازنده بازنویسی کنید و سپس برقراری سه شرط Mutual exclusion ، Progress و Bounded waiting را در الگوریتم خود بررسی کنید.

سوال چهارم)

دو پردازنده برای حل مسائل ناحیه بحرانی از روش‌های زیر استفاده کرده‌اند (متغیرهای L1 و L2 در هر دو مشترک هستند و مقدار Boolean دارند و در ابتدا به صورت تصادفی مقداردهی شده‌اند). هر کدام از سه شرط Mutual Exclusion, Progress, Bounded Waiting را بررسی کنید و توضیح دهید.

```
// P1
while (L1 != L2);

//Critical Section

L1 = !L2
```

```
// P2
while (L1 == L2);

//Critical Section

L1 = L2
```

سوال پنجم)

کلاس زیر که پیاده‌سازی سمافور است را کامل کنید و توضیح دهید هر بخش از کد که اضافه می‌کنید چگونه به حفظ سه شرط Mutual Exclusion, Progress, Bounded Waiting کمک می‌کند (فرض کنید که کلاس Process دو متد block و wakeup دارد).

```
class Semaphore {
    queue :Queue<Process>
    //other class Properties

    constructor Semaphore(initialValue: int){
    }

    wait(process: Process){
    }

    signal(){
    }
}
```