به نام خدا

دانشکده مهندسی کامپیوتر

**سیستم‌های عامل (۴۰۴۲۴) - گروه ۲**
نیم‌سال دوم سال تحصیلی ۱۴۰۱-۱۴۰۲
**استاد درس: دکتر رسول جلیلی**

تمرین سوم

مهلت تحویل: ۲۳:۵۹ روز چهارشنبه ۱۴۰۲/۰۳/۰۳

نکات و قواعد

۱. محل بارگذاری تمرین تا سه روز پس از مهلت تحویل باز خواهد بود. در طول ترم، برای تکالیف عملی و تئوری در مجموع می‌توانید از ۱۲ روز تاخیر مجاز به صورت ساعتی استفاده کنید و پس از آن به ازای هر روز ۲۵ درصد جریمه بر روی نمره‌ی کسب شده اعمال خواهد شد.

۲. لطفا حتما آداب‌نامه‌ی انجام تمرین‌های درسی را رعایت نمایید. در صورت استفاده از هر مرجعی و یا همفکری برای پاسخ به سوالات، مرجع مربوطه و یا نام همفکران را در پاسخ خود ذکر کنید.

۳. در صورتی که پاسخ سوالات را به صورت دست‌نویس آماده کرده‌اید، لطفا تصاویر واضحی از پاسخ‌های خود ارسال کنید. در صورت ناخوانا بودن پاسخ ارسالی، نمره‌ای به پاسخ ارسال شده تعلق نمی‌گیرد.

۴. فایل مربوط به پاسخ خود را به فرمت **OS_HW۳_StdNum_FirstName_LastName** نامگذاری کرده و ارسال نمائید.

۵. سوالات خود را می‌توانید به این آدرس ارسال کنید.

## سوال ۱ (۱۰ نمره)

میزبان یک میهمانی $N > 2$ نفر میهمان را به خانه دعوت کرده است. میزبان نمی‌خواهد چندین مرتبه در خانه را برای ورود میهمانان باز کند. $N$ میهمان برای یکدیگر صبر می‌کنند و به یک باره وارد خانه میزبان می‌شوند. میزبان و میهمانان با برنامه‌ی چند ریسمانی پیاده‌سازی شده‌اند. میزبان برای ورود همه مهمانان صبر می‌کند سپس تابع openDoor() را فراخوانی می‌کند و یک متغیر شرطی را سیگنال می‌دهد. مهمانان باید برای ورود $N$ نفر و باز شدن در صبر کنند و enterHouse() را فراخوانی کنند. کد میزبان در تکه کد زیر آورده شده است، تنها با استفاده از متغیرهای تعریف شده در کد میزبان، کد مهمان را بنویسید.

```
// Host
lock(m);
while (guest_count < N)
        wait(cv_host, m);
openDoor();
signal(cv_guest);
unlock(m);
```

## سوال ۲ (۱۵ نمره)

می‌خواهیم گذر ماشین‌ها از روی یک پل یک طرفه را مدیریت کنیم به صورتی که تا وقتی ماشین‌ها در حال عبور از پل از یک سمت هستند، به ماشین‌های سمت دیگر که قصد ورود به پل را دارند، اجازه عبور داده نشود. محدودیتی روی تعداد ماشین‌هایی که روی پل هستند وجود ندارد، اما می‌خواهیم مسأله گرسنگی را تا حدی حل کنیم. بدین منظور اگر ۵ ماشین از سمت شمال از پل رد شدند و تعدادی (یک یا بیشتر) ماشین در سمت جنوب قصد ورود به پل را داشتند، از ادامه عبور ماشین‌ها از سمت شمال جلوگیری کرده تا ماشین‌هایی که در سمت جنوب پل هستند بتوانند از پل عبور کنند. همین قانون برای سمت دیگر پل نیز به صورت برعکس برقرار است.

Semaphor یا Mutexهای لازم برای حل مسئله را تعریف کرده و دو تابع North و South را با استفاده از آن‌ها پیاده‌سازی کنید. با تابع North ماشینی که در شمال پل است از پل عبور می‌کند و با تابع South ماشین‌هایی که در سمت جنوب پل هستند از پل رد می‌شوند.

## سوال ۳ (۱۵ نمره)

برنامه زیر را در نظر بگیرید. در این کد، یک راه حل نرم‌افزاری برای مسئله انحصار متقابل برای دو پردازه پیشنهاد شده است. برقراری یا عدم برقراری چهار شرط لازم برای یک راه حل انحصار متقابل را برای این کد بررسی کنید. با توضیحات کافی نشان دهید کدام یک از این شرایط در مورد کد زیر صحیح است و با بیان مثال نقض نشان دهید کدام یک از شرایط اشتباه است.

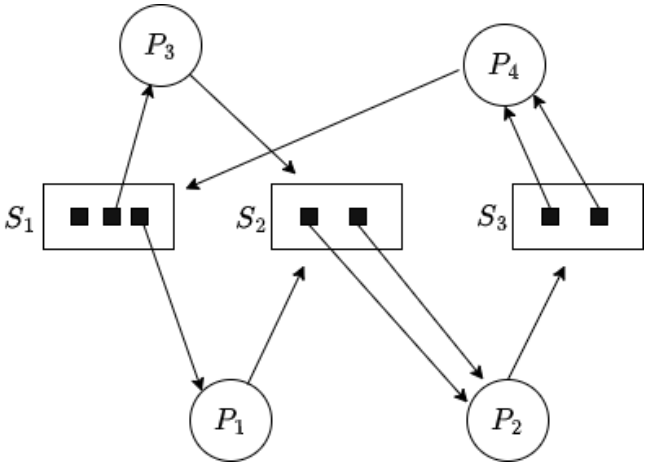
```
bool blocked[2];
int turn;

void P(int id)
{
    while(true)
    {
        blocked[id] = true;
        while (turn != id)
        {
            while (blocked[1-id])
                /* do nothing */
            turn = id;
        }
        /* critical section */
        blocked[id] = false;
        /* remainder */
    }
}

void main()
{
    blocked[0] = false;
    blocked[1] = false;
    turn = 0;
    parbegin(P(0), P(1)); // executed concurrently
}
```

## سوال ۴ (۱۵ نمره)

گراف تخصیص منابع[1] زیر را با ۳ منبع تجدید پذیر و ۴ پردازه در نظر بگیرید:




برای هر یک از پردازه‌ها مشخص کنید آیا در وضعیت بن بست قرار می‌گیرند یا خیر.

[1] Resource-Allocation Graph

## سوال ۵ (۱۵ نمره)

یک رستوران را در نظر بگیرید که فقط یک آشپز دارد. فرض کنید آشپز در هر لحظه فقط می‌تواند سفارش یک مشتری را آماده کند و تا زمانی که سفارش مشتری آماده نشده، سراغ سفارش بعدی نمی‌رود. با این‌حال، فرض کنید در هر لحظه، ممکن است یک مشتری به صورت تلفنی یا حضوری یک سفارش غذا ثبت کند. کد زیر را در نظر بگیرید. در این کد، تابع chef پیاده‌سازی روند آماده کردن غذا توسط آشپز است و تابع order تابعی است که برای ثبت سفارش جدید توسط مشتریان فراخوانی می‌شود. همچنین دو سمافور chef_done و customer را تعریف کرده‌ایم. با اضافه کردن توابع wait و signal در دو تابع ذکر شده، شرایطی را پیاده‌سازی کنید که آشپز زمانی که هیچ سفارشی وجود ندارد، غذایی آماده نکند و منتظر بماند و مشتری تا زمانی که هنوز سفارشش را دریافت نکرده تابع pay را فراخوانی نکند. همچنین مقادیر اولیه برای دو سمافور ذکر شده را مشخص کنید.

```
chef()
{
    while(true)
    {
        cook();
    }
}

order()
{
    pay();
}
```

## سوال ۶ (۱۵ نمره)

مسئله غذا خوردن فیلسوفان را با پنج فیلسوف در نظر بگیرید. فرض کیند راه حل زیر برای این مسئله ارائه شده است. آیا این راه حل به بن بست منجر می‌شود؟ به گرسنگی چطور؟ توضیح دهید.

```
void phl(int i)
{
    while(true)
    {
        think();
        take_fork(i); // Take forks number i and (i+1)%n if available, otherwise wait
        eat();
        put_fork(i); // Put down the forks
        put_fork((i+1) % n);
    }
}
```

## سوال ۷ (۱۵ نمره)

حالت لحظه‌ای یک سیستم را به صورت زیر در نظر بگیرید. ابتدا ماتریس نیاز را تشکیل دهید و سپس با توضیحات کافی مشخص کنید که سیستم در حالت امن قرار دارد یا خیر.

| | *Allocation* | *Max* | *Available* |
|---|---|---|---|
| | *A B C D* | *A B C D* | *A B C D* |
| $P_0$ | 0 0 1 2 | 0 0 1 2 | 1 5 2 0 |
| $P_1$ | 1 0 0 0 | 1 7 5 0 | |
| $P_2$ | 1 3 5 4 | 2 3 5 6 | |
| $P_3$ | 0 6 3 2 | 0 6 5 2 | |
| $P_4$ | 0 0 1 4 | 0 6 5 6 | |

موفق باشید