به نام یکتای مهربان

امتحان پایان ترم سیستم عامل
نیمسال اول ۱۴۰۰
دانشکده برق و کامپیوتر

ارزش امتحان: ۶۰
نام و نام خانوادگی دانشجو:

مدت زمان امتحان: ۱۲۰ دقیقه
استاد درس: زینب زالی

۱- همگام‌سازی (synchronization) (۲۰ نمره)

الف) راه حل زیر برای مسیله خوانندگان نویسندگان (اولویت با نویسندگان) ارایه شده است. با ذکر دلیل توضیح دهید آیا انحصار متقابل لازم برای مسیله خوانندگان نویسندگان رعایت شده است؟ همچنین با ذکر دلیل بیان کنید آیا امکان ایجاد بن بست و گرسنگی در این راه حل وجود دارد یا نه.

```
int writer_cnt = 0;
semaphore  reader = 1;
semaphore  cnt_mutex = 1 ,  w_mutex = 1;
```

reader	**writer**
If (writer_cnt>=1) wait(reader) /* CS: reading is performed */	wait(cnt_mutex); writer_cnt ++; if(writer_cnt==1) wait(reader) signal(cnt_mutex); wait(w_mutex); /* CS: writing is performed */ signal(w_mutex); wait(cnt_mutex); writer_cnt --; if(writer_cnt==0) signal(reader); signal(cnt_mutex);

ب) در یک سیستم آموزش مجازی کمک درسی، هر معلم میتواند در صورت تمایل، اعلام تشکیل کلاس مجازی دهد، از طرفی دانش آموزان متقاضی تشکیل کلاس هم می‌توانند در کلاسها شرکت کنند. در صورتی یک کلاس تشکیل می‌شود که یک معلم متقاضی موجود باشد و دانش‌اموزان متقاضی هم دقیقاً ۱۰ نفر باشند. هر معلم متقاضی تابع createClass و هر دانش‌آموز تابع joinClass را در threadهای مختلف اجرا می‌کند. اما تا قبل از اینکه معلمی کلاس تشکیل نداده باشد یا تعداد دانش‌آموزان به ۱۰ نرسیده باشد کلاس آغاز نمی‌گردد (startClass) و معلم یا دانش‌آموز تا فراهم‌شدن شرایط به حالت انتظار می‌روند. اگر تعداد دانش‌اموزان بیش از ۱۰ باشد، بقیه دانش‌آموزان در کلاس‌های بعدی(در صورت وجود معلم) قرار می‌گیرند. توابع createClass و joinClass رابا استفاده از سمافور و یا متغیر شرطی تکمیل کنید (در سودوکدهایی که می‌نویسید startClass در محل مناسب فراخوانی شود متغیر و سمافورهای لازم به درستی **تعریف و مقداردهی اولیه** شوند)

joinClass	**createClass**
//your code startClass() //yourCode	//your code startClass() //your code

---

۲- بن‌بست (۲۰ نمره)

الف) کدهای دو تابع f1 و f2 در زیر آمده این دو تابع به صورت مشترک از دو منبع استفاده می‌کنند هر یک از این دو منبع در هر لحظه فقط توسط یکی از توابع می‌تواند استفاده شود. با دو روش مختلف <نقض circular wait> و <نقض no preemption>، کدهای زیر را به نوعی **تغییر دهید یا بازنویسی کنید** که از ایجاد بن‌بست (deadlock) جلوگیری شود.

semaphore s1=1, s2=1;

f1	f2
wait(s1) wait(s2) //CS signal(s1) signal(s2)	wait(s2) wait(s1) //CS signal(s2) signal(s1)

ب) در یک سیستم، موجودیهای ۳ منبع به صورت A=10, B=4, C=8 است. اگر وضعیت تخصیص منابع در یک حالت سیستم به صورت زیر باشد، سپس همزمان p1 و p2 هر کدام به صورت جداگانه ۲ واحد C درخواست دهند، آیا سیستم با توجه به الگوریتم بانکداران می‌تواند درخواست آن‌ها را پاسخ دهد؟ (پاسخ با ذکر کامل روند حل مسیله با الگوریتم بانکداران)

process	Max need			allocation		
	A	B	C	A	B	C
p1	10	3	4	5	3	2
p2	0	2	3	0	0	1
p3	5	0	3	2	0	0
p4	3	0	3	2	0	1

۳- مدیریت حافظه (۲۰ نمره)

الف) در یک سیستم مدیریت حافظه از صفحه‌بندی (paging) با اندازه صفحه (page) ۸ کیلوبایت استفاده می‌شود و فضای آدرس منطقی(virtual) هر پروسس، یک مگابایت و حجم حافظه فیزیکی موجود ۴ مگابایت است، (۱) جدول صفحه (page_table) در این سیستم دارای چند سطر است؟ (۲) با توجه به سطرهای ۱۰ تا ۱۳ جدول صفحه که در شکل با اعداد دسیمال (ده دهی) آمده است، آدرس منطقی **Hex**18002، معادل چه آدرس فیزیکی در فرمت Hex است؟ (مراحل بدست آوردن جوابها را بنویسید.) (۳) اگر بخواهیم حافظه را بهینه‌تر استفاده کنیم و یا سرعت دسترسی به حافظه بالاتری داشته باشیم اندازه صفحه را بزرگ‌تر کنیم یا کوچکتر؟ شرح دهید

100
220
110
500

ب) یک روش page replacement به صورت زیر پیشنهاد شده است:

هر فریم، دارای یک بیت رفرنس است. به صورت پیش‌فرض این بیت برای همه فریمها صفر است. هرگاه به یک صفحه رجوع می‌شود، بیت رفرنس فریم متناظرش یک می‌شود. یک اشاره گر به صورت چرخشی روی فریمها حرکت می‌کند. هرگاه بخواهیم صفحه‌ای را برای جایگزینی (replacement) انتخاب کنیم بیت رفرنس فریمی که اشاره‌گر بدان اشاره می‌کند را چک میکنیم اگر این بیت صفر بود، صفحه مربوطه برای جایگزینی انتخاب می‌شود. اما اگر این بیت یک بود بیت تبدیل به صفر می‌شود و اشاره‌گر جلو می‌رود و به همین ترتیب صفحات بعدی برای جایگزینی بررسی می‌شوند تا به اولین فریمی که بیت رفرنسش صفر است برسیم.

تعداد page fault را برای این الگوریتم روی رشته رفرنس زیر (reference string)، از چپ به راست با ۴ فریم بدست آورید:

7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

این الگوریتم را از نظر تعداد page fault، سربار زمانی و حافظه ای اجرا با الگوریتم LRU مقایسه کنید (متناسب با جواب، علت را توضیح دهید)