بسمه تعالی

تمرین شماره ۵

سیستم‌های عامل

محسن کربلائی امینی، ۹۸۲۴۲۱۲۸

آذر ۱۴۰۲

سوال ۱:

۱. درست است. با این فرض حلقه‌ای برای آزاد شدن منابع شکل نمی‌گیرد بنابراین هیچ‌گاه به بن‌بست نمیرسیم. در صورت قرارگیری تعدادی از یک گروه در کنار یکدیگر، به محض رسیدن به یک نفر از گروه دیگر، حتما یکی از گروه اول می‌تواند شروع به خوردن کند. بنابراین هیچ‌گاه شرط circular wait برقرار نمیشود و deadlock رخ نمی‌دهد.

۲. نادرست. از آنجایی که از هر منبع (چنگال۱و چنگال۲و ...چنگال ۵) تنها یک واحد داریم، امکان ندارد چنین اتفاقی بیوفتد. در واقع در این حالت دو نفر چنگال اولشان یکی خواهد بود و این که این واحد به هر دوی آن‌ها برسد ممکن نیست و بنابراین حداقل یک نفر چنگالی برنخواهد داشت. **اگر چه به طور کلی امکان وقوع بن‌بست وجود ندارد.**

۳. نادرست. از شروط اولیه بن‌بست که circular wait می‌باشد، موجود نیست و هیچ‌گاه بن‌بست اتفاق نمی‌افتد.

۴. نادرست. از شروط اولیه بن‌بست که circular wait می‌باشد، موجود نیست و هیچ‌گاه بن‌بست اتفاق نمی‌افتد.

سوال ۲:

الف)

	need			Available		
	A	B	C	A	B	C
P۰	۳	۵	۶	۰	۲	۳
P۱	۵	۳	۳			
P۲	۲	۱	۳			
P۳	۲	۵	۷			
P۴	۱	۲	۳			

در این حالت مقدار available حاصل شده برای منبع A نمی‌تواند به هیچ فرایندی تخصیص داده شود. بنابراین تمام فرایندها منتظر خالی شدن منبع A می‌مانند. این حالت safe نیست و بنابراین پس از انجام این درخواست، وقوع بن‌بست قطعی ست.

ب)

۱. قابل انجام است.

	need			Available		
	A	B	C	A	B	C
P۰	۳	۵	۶	۱	۲	۳
P۱	۵	۱	۳			
P۲	۲	۱	۳			
P۳	۳	۵	۷			
P۴	۱	۲	۳			

$< ۱,۲,۳ > -p۴-< ۲,۳,۳ > -p۲-< ۵,۵,۳ > -p۱-< ۷,۷,۶ > -p۰-< ۷,۸,۸ > -p۳-< ۸,۸,۱۰ >$

۲. قابل انجام نیست. حالت حاصل شده، safe نیست و نمیتوان هیچ‌کدام از فرایندها را اجرا کرد چرا که همه فرایندها به نمونه‌ای از منبع A نیاز دارند که در دسترس نمی‌باشد.

۳.

	need			Available		
	A	B	C	A	B	C
P۰	۳	۵	۶	۰	۳	۲
P۱	۵	۳	۳			
P۲	۱	۰	۲			
P۳	۳	۵	۷			
P۴	۱	۲	۳			

سوال ۳:

۱. درست است اما در این حالت حتما بن‌بست رخ داده است. چرا که فرایندهای داخل دور همه منتظر یکدیگر هستند تا نمونه‌ای را یکی از آن‌ها خالی کند.

۲. نادرست. در صورت خاتمه یک فرایند تمامی منابع آن فرایند آزاد خواهد شد.

۳. درست. این پارامتر می‌تواند در این تصمیم نقش داشته باشد چرا که منابعی که فرایندهای بیشتر را سیر می‌کند آزاد می‌کند. البته این مورد ممکن است در پیاده‌سازی‌های مختلف لزوما مورد استفاده قرار نگیرد.

۴. نادرست. هر وضعیت unsafe لزوما به این معنی نیست که بن‌بست رخ خواهد داد. در الگوریتم avoidance از حالت‌های بن‌بست دوری میکنیم تا هیچ‌گاه بن‌بست رخ ندهد.

۵. نادرست. در هنگام بن‌بست، تمامی فرایند منتظر آزاد شدن منابع فرایند دیگری هستند که خود آن فرایندها منتظر دیگری هستند و به این ترتیب در یک حلقه قرار میگیرند که همه منتظر همدیگر خواهند بود.

۶. درست. در این گراف حلقه وجود دارد. $p۴ > R۱ > P۲ > R۲$

۷. نادرست. پس از آزاد شدن منبع R۱ توسط فرایند P۱، منبع مورد نیاز P۴ تامین خواهد شد. و پس از انجام شدن P۴ یا P۳ هم منبع مورد نیاز P۲ آزاد خواهد شد.

سوال ۴:

$Available: ۳$

$Assignment\ Sequence: ۳ - P۴ - ۴ - P۳ - ۶ - P۲ - ۷ - P۱ - ۹$

سوال ۵:

a) خیر

$< ۳,۲,۲ > - P۲ - < ۴,۲,۳ > - P۳ - < ۵,۳,۴ > - P۰ - < ۷,۴,۴ > - P۱ - < ۷,۶,۶ > - P۴ - < ۸,۶,۸ >$

b) خیر

$< ۳,۲,۲ > - P۲ - < ۴,۲,۳ > - P۳ - < ۵,۳,۴ > - P۱ - < ۵,۵,۶ > - P۰ - < ۷,۶,۶ > - P۴ - < ۸,۶,۸ >$

c) خیر

$< ۳,۲,۲ > -P۳-< ۴,۳,۳ > -P۲-< ۵,۳,۴ > -P۱-< ۵,۵,۶ > -P۰-< ۷,۶,۶ > -P۴-< ۸,۶,۸ >$

d) بله

$< ۳,۲,۲ > -P۲-< ۴,۲,۳ > \rightarrow deadlock$

سوال۶:

a) انواع روش‌های حل بن‌بست:

- Ostrich Algorithm
- Detection
- Prevention
- Avoidance

b) تفاوت‌های روش Prevention و Avoidance:

- **Prevention:** در این روش از ایجاد بن‌بست در سیستم پیش‌گیری می‌کنیم. مدیریت درخواست‌ها و شرایط ۴ گانه که موجب ایجاد بن‌بست می‌شوند به طوری که هیچ‌گاه با بن‌بست روبه‌رو نشویم. مثلا در شرط Hold and Wait باید تضمین شود که اگر یک فرایند درخواست منبعی را دارد، نباید منبعی دیگری را در اختیار داشته باشد.
- **Avoidance:** در این روش به گونه‌ای عمل می‌کنیم که از ایجاد بن‌بست دوری کنیم. برای این کار حالت‌های امن را تعریف کردیم و در صورتیکه قبول یک درخواست منبع از طریف یک فرایند، سیستم را در حالت unsafe قرار دهد، وارد آن حالت نشده و آن درخواست رد خواهد شد.

c) روش‌های ارائه شده برای هر مشکل:

- **Mutual Exclusion:** این شرط برای منابع قابل اشتراک‌گذاری مورد نیاز نیست(برای مثال فایل‌های فقط خواندنی). اما باید برای منابعی که قابل اشتراک‌گذاری نیستند، حتما hold کرد.
- **Hold and Wait:** باید تضمین شود که اگر یک فرایند درخواست منبعی را دارد، نباید منبعی دیگری را در اختیار داشته باشد. از مشکلات این مورد می‌توان به کم بودن بهره‌برداری از منابع، احتمال starvation و لزوم اعلام از پیش فرایندها برای تمامی درخواست هایی که دارند، که این پیش‌بینی می‌تواند سخت باشد.

- **No Preemption:** اگر یک فرایند درخواست منابعی را می‌دهد که امکان تخصیص در آن زمان نیست، باید تمام منابعش را آزاد کند.
- **Circular Wait:** تحمیل یک سفارش کلی از همه انواع منابع، و مستلزم این است که هر فرآیند درخواست منابع در یک افزایش ترتیب شمارش

سوال ۷:

a) دو روش برون‌رفت از بن‌بست:

- **اتمام فرایندهای درون بن‌بست:** یکی یکی فرایندها را قطع کرده تا بن‌بست رفع شود. معایب این روش می‌تواند این باشد که بعضا فرایندهایی که هزینه زیادی برای آن‌ها صرف شده، از بین می‌روند و پردازش مجدد آن‌ها می‌تواند هزینه‌بر باشد. از طرف دیگر این روش می‌تواند برون‌رفت از بن‌بست را به سرعت و سادگی انجام دهد و از starvation دوری می‌کند.
- **گرفتن منابع یک فرایند:** انتخاب قربانی، بازگشت به یک حالت safe. در این روش ممکن است starvation رخ دهد به این صورت که فرایندهای یکسانی هر بار به عنوان قربانی شناخته شوند.

در استفاده از هر دو روش فاکتورهایی مهم هستند که در انتخاب فرایندی که قربانی می‌شود اثر گذار است. مانند اولویت فرایند، زمانی که تا الان برای پردازش آن هزینه شده و زمان پردازشی که از آن باقی مانده است، میزان منابع آن، و میزان منابعی که نیاز دارد تا اتمام کامل داشته باشد.

b) این مفهوم به معنی بازگشت به حالتی قبل که مشکلی که الان بوجود آمده، وجود نداشت و سیستم دچار بن‌بست نبوده است. به طور کلی در این متن، بازگشت به معنی بازگشت به یک حالت safe بعد از ورود به یک حالت unsafe معنی می‌شود.

c) به این دلیل بعد از این اتفاق طبعا آن فرایند منابعی ضروری که نیاز داشته را از دست داده و نمی‌تواند به درستی اجرا شود. بعد از انجام rollback، فرایند باید مجددا از آن حالت شروع به اجرا کند.

d) باید مطمئن شد که یک فرایند تعداد محدودی rollback شود، برای مثال می‌توان برای هر فرایند یک هزینه حساب کرد که در صورت rollback آن فرایند، سیستم متحمل خواهد شد. سپس در این هزینه تعداد rollback های آن فرایند را نیز لحاظ کنیم.