پاسخنامه تکلیف دوم سیستم عامل
ترم اول ۱۴۰۰

سوال 1:

الف)

استفاده از کوانتوم زمانی کوچک به علت افزایش تعداد پروسه هایی که در یک مدت زمان مشخص، روی پردازنده قرار میگیرند برای افزایش Responsiveness مناسب است. زمانی که صف پروسه های آماده پر از پروسه های تعاملی باشد استفاده از کوانتوم زمانی کوچک منطقی تر است. زیرا این پروسه ها نیاز به response time پایین دارند و از طرفی معمولا burstهای کوچک دارند.

ب)

استفاده از کوانتوم زمانی بزرگ به علت کاهش تعداد تعویض متن های متوالی برای افزایش فاکتورهای بهره‌وری پردازنده و افزایش توان عملیاتی (Throughput) مناسب است.

ج)

برای افزایش Responsiveness باید از کوانتوم زمانی کوچکتری استفاده کرد تا برای پروسه های کوچک جدید فرصت اجرا باشد و زمانبند نیز بتواند آنها را در مدت زمان کوتاهتری (با زمان انتظار کمتری) انتخاب کند. معمولاً سیستم هایی که جهت استفادهٔ روزمره ساخته میشوند از کوانتوم زمانی کوچکتری استفاده میکنند تا پروسه های تعاملی سریعتر اجرا شوند.

برای افزایش توان عملیاتی و بهره‌وری پردازنده باید از کوانتوم زمانی بزرگتری استفاده کرد تا پروسه های بزرگ و غیرتعاملی بدون نیاز به تعویض متن های متعدد اجرا شوند. معمولاً سیستم هایی که جهت اجرای کارهای دسته‌ای (Batch Jobs) استفاده میشوند از کوانتوم زمانی بزرگتری استفاده میکنند.

د)

برای سیستم هایی که جهت اجرای کارهای طولانی و سنگین که چند تعامل کوتاه از کاربر نیز نیاز دارد کوانتوم زمانی کوچک و بزرگ هردو منطقی خواهد بود. در این سیستم ها در مدت زمانی که نیاز به تعامل با کاربر کاهش می یابد، کوانتوم زمانی بزرگتر میشود تا بهره‌وری پردازنده و توان عملیاتی افزایش یابد و سپس در صورت نیاز (افزایش نیاز به تعامل با کاربر) کوانتوم زمانی کوچکتر میشود.

سوال 2:

FCFS

P1	P2	P3	P4	P5
2	3	11	15	20

TT= [(2-0)+(3-1)+(11-2)+(15-4)+(20-4)]/5

WT= [(0-0)+(2-1)+(3-2)+(11-4)+(15-4)]/5

SJF

P1	P2	P3	P4	P5
2	3	11	15	20

TT= [(2-0)+(3-1)+(11-2)+(15-4)+(20-4)]/5

WT= [(0-0)+(2-1)+(3-2)+(11-4)+(15-4)]/5

SRF

P1	P2	P3	P4	P5	P3
2	3	4	8	13	20

TT= [(2-0)+(3-1)+(20-2)+(8-4)+(13-4)]/5

WT= [(0-0)+(2-1)+((3-2)+(13-4))+(4-4)+(8-4)]/5

RR

P1	P2	P3	P4	P5	P3	P4	P5	P3	P5	P3
2	3	5	7	9	11	13	15	17	18	20

TT= [(2-0)+(3-1)+(20-2)+(13-4)+(18-4)]/5

WT= [(0-0)+(2-1)+((3-2)+(9-5)+(15-11)+(18-17))+((5-4)+(11-7))+((7-4)+(13-9)+(18-15)]/5

سوال 3-1:

الف)
پروسهٔ CPU-Bound از این روش سود بیشتری میبرد. چرا که تا زمانی که به طور کامل انجام نشود، زمانبند مرتباً کوانتوم زمانی بیشتری به آن پروسه اختصاص میدهد. ضمناً اولویت آن پروسه نیز بیشتر میشود و الگوریتم آن را زودتر از پروسهٔ IO-Bound انتخاب میکند.
ب)
هدف از این سیاست اولویت دادن به پروسه های CPU-Bound طولانی در سیستم هایی که Interactive بودن اولویت نباشد است. هدف دیگر کاهش زمان از دست رفته به علت تعویض متن های متعدد هنگام اجرای پروسه های CPU-bound است که باعث افزایش درصد مصرف CPU می‌شود. همچنین احتمالا پروسسهای IO bound دارای burst های طولانی IO هستند و در مدت زمانی که منتظر IOهستند نیاز نیست در CPUاجرا شوند بنابراین در این فاصله پروسسهای CPU bound که اولویت بالاتری هم دارند زودتر زمان بندی میشوند و اگر تعداد پروسه های IO bound کم باشد آنها هم در حالت ready زمان انتظار زیادی نخواهند داشت

سوال 3-2:

الف) یک ماژول است که پس از انتخاب شدن یک پروسه توسط زمانبند آن را روی پردازنده قرار می‌دهد و مسیول عملیات context switch است.
ب) پروسه‌ها دوست دارند روی همان پردازنده‌ای که قبلا در آن اجرا شده‌اند اجرا شوند. زیرا احتمالاً داخل کش آن پردازنده اطلاعات آن‌ها وجود دارد و به ارتباط کمتری با ممووری نیاز می‌شود.
ج) زمان رانتایم یک پروسه است که نسبت به Niceness پروسه‌ها نرمالایز شده است. بنابراین برای پروسه ای با اولویت بالا یا nice پایین این مقدار، کمتر از مقدار زمان واقعی اجرای پروسس است و برای پروسه ای که اولویت پایین یا niceبالا دارد بیشتر از مقدار زمان واقعی اجرای پروسس است

سوال 4:

الف) در یک سیستم RR هر چه کوانتوم زمانی را کوچکتر بگیریم زمان پاسخ پروسسها کمتر میشود (بهبود میابد) در عوض به دلیل تعویض متنهای زیاد زمان زیادی از CPU صرف تعویض متن میشود و بنابراین CPU Utilization کاهش می یابد (بدتر میشود). در مقابل اگر کوانتوم زمانی را افزایش دهیم تعداد تعویض متنها کم شده بنابراین CPU Utilization افزایش می یابد اما زمان پاسخ هم افزایش پیدا می‌کند.

ب) با توجه به رابطه زیر، با توجه به اینکه مجموع زمان burst برای تعداد ثابتی پروسس، یک مقدار ثابت است و وابسته به الگوریتم زمان بند نیست، بنابراین متوسط زمان برگشت رابطه مستقیم با متوسط زمان انتظار دارد. پس اگر بخواهیم دنبال حالتی باشیم که متوسط زمان برگشت و ماکزیمم زمان انتظار در تضاد هم باشند باید به دنبال حالتی باشیم که مثلا متوسط زمان انتظار کاهش یابد در حالی که ماکزیمم زمان انتظار افزایش می یابد.

$$avg(TT) = \sum_i TT_i \ /n = \sum_i WT_i + Burst_i \ /n$$

فرض کنید دو پروسس در سیستم هستند که یکی **burst** بلند (مثلا ۵) و دیگری **burst**کوچک (مثلا ۲) دارد و هر دو باهم وارد سیستم میشوند اینجا اگر بخواهیم متوسط زمان انتظار کاهش یابد باید **SJF** را استفاده کنیم پس پروسس کوتاه را اجرا میکنیم حال فرض کنید به محض اتمام پروسس کوتاه، پروسس کوتاه دیگری برسد دوباره مجبوریم پروسس جدید را اجرا کنیم اگر این کار تکرار پیدا کند و سیاست ما

هربار این باشد که برای کاهش زمان انتظار (**SJF** از این نظر اپتیمال است) پروسس کوچک را انتخاب کنیم پروسس بزرگ تا اتمام این پروسسهای کوچک به تاخیر می افتد . اکنون ماکزیمم زمان انتظار مربوط به همین پروسس بلند است که با این کار مرتب افزایشش داده ایم اما در حالی که متوسط زمان انتظار را کمینه نگه داشته ایم.

سوال 5:
Priority: اگر همیشه قبل از اینکه نوبت پروسه اولویت پایین برسد یک پروسه اولویت بالا وارد شود، پروسه اولویت پایین دچار گرسنگی می‌شود.
SJF: اگر همیشه قبل از اینکه نوبت پروسه طولانی برسد یک پروسه کوتاه وارد شود، پروسه طولانی دچار گرسنگی می‌شود.

سوال 6:

RR: در این الگوریتم همهٔ پروسه ها به مدت زمان یکسان زمانبندی می‌شوند. یعنی بین پروسه های کوچک و بزرگ از نظر اختصاص زمان سیاست "برابری" اعمال میشود.
FCFS: در این الگوریتم ترتیب در صف قرار گرفتن پروسه ها مهم است. از این نظر پروسه های کوچکتر که کمی دیرتر از پروسه های بزرگ آمده اند باید تا اتمام اجرای پروسهٔ بزرگتر منتظر بمانند.
MLFQ: در این الگوریتم (اگر مطابق با مثال کتاب در نظر بگیریم) پروسه های کوچکتر اولویت بالاتری دارند و در اولین صف زمان بندی قرار میگیرند.