پرسش های زیر دارای نمره ی اضافی هستند و پاسخ به آنها ضروری نیست. در صورت تمایل به دریافت نمره ی اضافی به آنها به صورت کامل پاسخ بدهید.

(آ) فرض کنید در سیستم عامل فراخوانی وجود دارد که پس از فراخوانی آن توسط برنامه فرآیند برای مدتی مسدود شده و پس از آماده شدن داده ی مورد نظر آن داده توسط سیستم عامل با برنامه به اشتراک گذاشته می شود. در این صورت به سوالات زیر پاسخ دهید:

i. از میان دو شیوه ی صندوق پیام و اشتراک حافظه کدام روش برای این منظور مناسب تر است؟ پاسخ خود را توجیه کنید.
اگر سیستم‌عامل بخواهد چنین فراخوانی را داشته باشد از آنجایی که میتواند مسدود کردن و سپس ادامه کار را به راحتی انجام دهد، استفاده از حافظه اشتراکی گزینه مناسب‌تری است چرا که سرعت بیشتری دارد، پیچیدگی خاصی ندارد و احتیاجی به کپی کردن دیتا در فرآیند نیست. سیستم‌عامل پس از اینکه اجرای فرآیند را مسدود کرد، داده را درون حافظه اشتراکی قرار میدهد و آدرس آن را به فرآیند می‌دهد و اجرای آن را از سر میگیرد.

ii. استفاده از روش دیگر چه معایبی خواهد داشت؟
استفاده از انتقال پیام به دلیل سربارى که روی کرنل به‌دلیل system call ها دارد نمیتواند گزینه خوبی برای دیتاهایی با اندازه زیاد باشد و به دلیل اینکه دیتا باید از صف پیام‌ها به حافظه فرآیند منتقل شود تاخیر بیشتری نیز دارد.

iii. به جز اشتراک پیام و اشتراک گذاری حافظه آیا روش بهتری برای انجام این کار سراغ دارید؟ روش خود را به صورت کامل شرح دهید و بگویید چرا این روش از روش های یاد شده در بالا بهتر است.
روش دیگری که می‌توان برای این کار استفاده کرد، فایل‌های موقت هستند که بدین صورت که سیستم‌عامل یک فایل موقت ایجاد کرده و داده‌ها را در آن ذخیره می‌کند و آدرس آن را به فرآیند برمی‌گرداند. در این روش فرآیند می‌تواند از این فایل بخواند، در آن بنویسد و یا حتی آن را ذخیره کنید تا بعدا از این فایل دوباره استفاده کند. این کار پیچیدگی کمتری از دو روش بالا دارد و سرعت آن نیز مناسب خواهد بود.

(ب) شما یک برنامه چند نخی برای پردازش داده های ورودی طراحی کرده اید که از سه نخ مختلف برای پردازش داده ها استفاده می کند. نخ اول باید داده ها را از فایل ها بخواند، نخ دوم داده ها را پردازش کرده و نخ سوم نتایج را ذخیره کند. پس از اجرا، متوجه می شوید که سرعت پردازش کندتر از حد انتظار است و برخی نخ ها به طور غیرضروری منتظر نخ های دیگر می مانند.

(آ) مشکل اصلی در نحوه زمان بندی نخ ها چیست؟
مشکلی اصلی در این است که زمانبندی نخ‌ها به صورتی است که وقتی داده خوانده نشده است، CPU به نخ پردازشی اختصاص داده میشود درحالی که چیزی برای پردازش وجود ندارد یا به نخ ذخیره سازی اختصاص داده میشود درحالی که چیزی برای ذخیره سازی وجود ندارد پس مدت زمانی را که میتوانست به نخی اختصای یابد که میتواند کار مفیدی انجام دهد به نخی اختصاص می‌یابد که باید منتظر بماند.

(ب) چگونه زمان بندی مناسب نخ ها می تواند به بهبود عملکرد برنامه کمک کند؟
اگر زمانبندی نخ‌ها به درستی انجام شود، نتیجه پردازش‌ها به صورت جریانی از داده‌ها ذخیره می‌شود و احتیاجی نیست که کاربر منتظر پایان تمام برنامه باشد تا از نتایج استفاده کند ولی این اتفاق نیازمند زمان‌بندی درست اجرای نخ‌ها است تا زمان کلی اجرای برنامه بیشتر از چیزی که باید نشود چرا که این موضوع باعث هدر رفتن وقت کاربر میشود.

(ج) در این سناریو، چه روش هایی برای جلوگیری از Context Switching غیرضروری و بهینه سازی زمان بندی نخ ها وجود دارد؟
در این سناریو باید از مکانیزم‌هایی استفاده کنیم که همگام‌سازی این نخ‌ها را به ارمغان می‌آورند. در این مکانیزم‌ها تا زمانی که داده‌ها برای پردازش به اندازه کافی نباشند، نخ پردازشی اجرا نمی‌شود و یا تا زمانی که چیزی پردازش نشده است، نخ ذخیره سازی اجرا نمی‌شود. اگر این مکانیزم‌ها را به علاوه اینکه برای هر نخ زمان اجرای مناسبی را در هر زمانی بین Context Switch در نظر بگیریم، می‌تواند از Context Switch های غیر ضروری جلوگیری کنیم و زمان اجرای برنامه را بهینه کنیم.

(د) اگر برنامه شما روی یک سیستم چند هسته ای اجرا شود، چه تفاوت هایی در نحوه زمان بندی نخ ها وجود خواهد داشت؟ چگونه می توان از هسته ها به طور بهینه استفاده کرد؟

چون برنامه رو یک سیستم چند هسته‌ای اجرا میشود دیگر احتیاج نداریم که بین نخ‌ها جابه‌جا شود اما همچنان به مکانیزم‌های همگام‌سازی نیاز داریم تا بتوانیم نخ‌ها را به صورت همزمان اجرا کنیم.

(ج) بررسی کنید آیا ممکن است که فرآیندی در یک بازه زمانی یتیم و سپس در بازه ای دیگر تبدیل به فرآیند زامبی شود؟ برعکس این سناریو محتمل است؟ (تشریح کنید و مثال بزنید)

بله این اتفاق ممکن است بیفتد برای مثال فرآیندی را درنظر بگیرید که terminate شده است درحالی که فرزندش هنوز اجرا می‌شود، این فرزند یتیم شده است و پس از اتمام اجرا ممکن است فرآیند دیگری که در مدیریت این فرآیند را برعهده گرفته است در حالت مسدود باشد و فرآیند فرزند تبدیل به زامبی شود.

برعکس این اتفاق ممکن نیست چرا که فرآیند زامبی، فرآیندی است که اجرای آن به اتمام رسیده است ولی فرآیند پدر هنوز به آن رسیدگی نکرده است پس حتی اگر پدر این فرآیند نیز terminate شود، سیستم‌عامل به فرآیند فرزند رسیدگی و آن را از لیست فرآیندها حذف می‌کند.