پاسخ سوال 1:

بن بست رخ نمی‌دهد.
منبع پایین را برابر با R0، منبع بالا سمت چپ را R1 و منبع بالا سمت راست را R2 در نظر بگیرید.
ابتدا می توان به پردازه P2 اجازه داد تا منبع مورد نیاز خود از R0 را بگیرد و با اتمام کار خود، منابع گرفته شده از R0 و R2 را پس دهد.
سپس پردازه P0 منابعی که اکنون از R0 و R2 آزاد شده را در اختیار می گیرد و با اتمام کار خود، منابع خود را پس می دهد. با آزاد شدن منبع R1، اکنون پردازه P1 می تواند منبع R1 را در اختیار بگیرد و با اتمام کار خود، منابع R1 و R2 را آزاد کند. اکنون پردازه P3 میتواند منابع مورد نیاز خود از R2 را بگیرد و کار خود را تمام کند. به همین دلیل بن بست رخ نمی‌دهد.
دنباله:P3 <- P1 <- P0 <- P2

پاسخ سوال 2: از الگوریتم بانکـدار استفاده می‌کنیم:

1) Let Work and Finish be vectors of length 'm' and 'n' respectively.
Initialize: Work = Available
Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both
a) Finish[i] = false
b) Need$_i$ <= Work
if no such i exists goto step (4)

3) Work = Work + Allocation[i]
Finish[i] = true
goto step (2)

4) if Finish [i] = true for all i , then the system is in a safe state

در ابتدا، مقادیر موجود در آرایه work برابر است با [1,1,2,3] و آرایه Finish به ازای تمام پردازه ها مقدار false دارد.

در میان پردازه ها یکی از پردازه‌هایی که شروط 2 را داشته باشد انتخاب می کنیم، مثلا P0

پس از اجرای مرحله 3، work برابر با [2,1,2,7] و آرایه Finish برابر با

[true, false, false, false, false] است.

سپس پردازه P2 را انتخاب می کنیم. پس از اجرای مرحله 3، work برابر است با [6,6,4,8] و Finish برابر است با [true, false, true, false, false].

سپس پردازه P3 را انتخاب می کنیم. پس از اجرای مرحله 3، work برابر است با [9,9,10,8] و Finish برابر است با [true, false, true, true, false].

در انتها ارایه Finish به ازای تمام پردازه ها برابر با true می شود. در نتیجه میتوان به درخواست های پردازه ها پاسخ داد. در نتیجه در حالت امن قرار داریم.

پاسخ سوال 3:

الف) حداقل منبع مورد نیاز برای اینکه بن بست رخ ندهد، برابر با 7 خواهد بود. زیرا بن بست در حالتی می تواند رخ بدهد که هر پردازه تعداد یکی کمتر از منابع مورد نیاز خود را در اختیار داشته باشد که با توجه به صورت سوال این مقدار برابر 6 خواهد بود. پس اگر یک منبع بیشتر از این داشته باشیم، حداقل یک پردازه می تواند در این شرایط کار خود را تمام کند و با آزاد کردن منابع خود، منابع دیگر پردازه ها را تامین کند.

ب) اگر حداکثر سه منبع داشته باشیم، حتما بن بست رخ خواهد داد، زیرا یکی از پردازه ها برای اجرای خود به چهار منبع نیاز دارد. اما اگر تعداد منابع بیشتر داشته باشیم، بن بست می تواند رخ بدهد یا ندهد.

پاسخ سوال ۴:

الف)یک Page table داریم که در حافظه قرار دارد. برای دسترسی به آن 50ns زمان نیاز داریم، از طرفی آدرس های درون Page table به خانه هایی در حافظه اشاره می‌کنند (به اصطلاح Page table of page table داریم)، در نتیجه 50ns هم اینجا مصرف می شود و در حالت کلی برای دسترسی به داده یا دستور موردنظر به 100ns زمان نیاز داریم.

ب)طبق رابطه زیر

$$EAT = h \times \alpha + (1 - h) \times 2\alpha$$

با توجه به اینکه h برابر 0.75 می باشد و همچنین آلفا برابر 2 نانوثانیه می باشد و جای 2آلفا، 100ns خواهد بود، خواهیم داشت:

*EAT = 0.75 * 2ns + (0.25) * 100ns = 1.5ns + 25ns = 26.5ns*

پاسخ سوال ۵:

ابتدا میزان خالص حافظه مورد نیاز را محاسبه می‌کنیم.

400 * 0.83 = 332MB

چون در حال استفاده از سیاست first fit هستیم، به ازای استفاده از هر N بلاک حافظه، N/2 بلاک اضافه هم تلف می‌شود در نتیجه میزان کل حافظه مورد نیاز برابر می‌شود با:

332 * 1.5 = 498MB

پاسخ سوال ۶:

الف)

[1] FIFO, Page faults = 8
39215554440000009
39211155544444403
XXX92221115555554
XXX39992221111115
XXXX3339992222221

[1] LRU, Page faults = 11
39215324910015129
X3921532491101512
XX392153249990051

XXX39215324449905
XXXX3991532224490

[1] Optimal, Page faults = 7
39215555555555555
X3921111111111111
XX392222222222222
XXX39999999999999
XXXX3334440000000

[2] FIFO, Page faults = 8
47700003319944447
X4477770031199994
XXX44447703311119
XXXXXXX4470033331
XXXXXXXXX47700003

[2] LRU. Page faults = 9
47704073319740347
X4470407731974034
XXX47740073197403
XXXXXXX4407319770
XXXXXXXXX40031999

[2] Optimal, Page faults = 6
47700003319999999
X4477770033333333
XXX44447700000000
XXXXXXX4477777777
XXXXXXXXX44444444

ب)

در مجموع: ۱۹۶ بار Page Fault رخ می‌دهد.

در ابتدا به ازای هر صفحه یک Page Fault خواهیم خورد (۱۰۰)

در مرحله بعدی، به دلیل استفاده از سیاست FIFO، چهار صفحه آخر را داریم، در نتیجه این ۴ صفحه Page Fault نمی‌خورند، اما برای ۹۶ صفحه بعدی مجددا Page Fault خواهیم داشت.