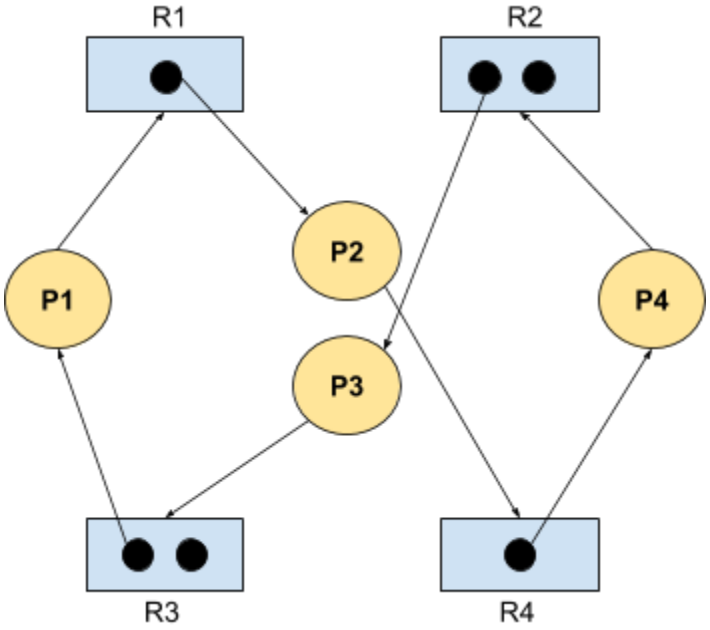
CS143a Spring 2020 Name: _____
Homework 3

*Please do not modify the structure of this document. Due:* ***5/25 at 11:59pm.***

---

**1) Deadlocks [27 pts]**

**a) [6]** A week has passed since the midterm, and Alice has not touched OS that much (other than attending lectures and discussions, of course). She wants to catch up on everything that she might have missed, and so she decides to binge-watch all the lectures like a Netflix series. However, this causes Alice to sleep at her desk a lot, giving Bob a chance to cause a deadlock on Alice's computer. He starts up some processes, has them request some resources as depicted below in the following resource allocation graph, and then escapes the room. Seconds later, Alice wakes up and notices that Bob has done something to her computer. Fortunately, Alice notices that Bob left his drawing of the resource allocation graph, and she decides to use that to see what Bob has done. Mark **X** in the appropriate boxes below answering whether there is a cycle and whether the system is in a deadlock.




| | Yes | No |
|---|---|---|
| **Has a cycle?** | X | |
| **In a deadlock?** | | X |

**b) [6]** The Start of Alice's Revenge. Alice decides that after all of Bob's pranks, now is the time to get revenge. Alice takes one of Bob's toasters (of which there are many), and sees that it is executing 3 user processes P1, P2, and P3. Alice learns that each process requires a certain number of instances of a resource type R - P1 requires a maximum of 2 instances of resource type R, P2 requires a maximum of 4 instances of resource type R, P3 requires a maximum of 5 instances of resource type R. Alice has an idea for what to do, but she needs to first figure out the minimum number of units of R needed so that a deadlock does not occur. Calculate this number, and show all work.

Number of Units of R: ____9____
Explanation:
$(R_{P1}-1) + (R_{P2}-1) + (R_{P3}-1) = 1 + 3 + 4 = 8$
This is the maximum number of units of R that ensures a deadlock occurs.
If we have an additional R, any one of P1, P2, or P3 can be completed and will return the resources which are enough for other processes to be completed.
So, 8+1 = 9

**c) [15]** After figuring the above out, Alice decides that she needs to take a break. She decides to go online and play a game of *OS Monopoly: The Deadlock Avoidance Edition* with 4 other random players online. In this game, 4 people are given an ID and act as a process (therefore, player 0 is P0, player 1 is P1, etc.); they move around the board to obtain sufficient resources in order to complete a given task. Each square on the board represents a set of resources that a player can request, but all resources are controlled by the last player - the "bank" player - and may only be claimed if it leaves the bank in a "safe state". Suppose that in Alice's game, there are a total of 3 instances of R0, 3 instances of R1, 1 instance of R2, and 2 instances of R3; this will be represented with a vector (3,3,1,2). In addition, processes must declare their maximum asks - P0 declares it needs a maximum of (1,0,1,1) resources -similarly P1 declares (1,1,1,0) resources, P2 declares (1,1,1,1) resources, and P3 declares (1,0,1,1) resources. After a few minutes, suppose that Alice (playing as P0) has obtained (1,0,1,0), P1 has obtained (0,1,0,0), P2 has obtained (1,1,0,0), and P3 has obtained (1,0,0,1). Alice remembers that a table like the one used for Banker's Algorithm might be useful here.

| | **Allocation** | | | | **Need** | | | | **Available** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R0 | R1 | R2 | R3 | R0 | R1 | R2 | R3 | R0 | R1 | R2 | R3 |
| P0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| P1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | | | | |
| P2 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | |
| P3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | | | | |

i) Alice wants to know if the bank player made any mistakes (i.e., if the bank player left the bank in an unsafe state), because doing so could leave the players in deadlock. Mark an **X** in the appropriate box indicating whether the players are in a deadlock. If yes, list all additional resource vectors that could lead to avoid the deadlock state (e.g., list out (1,0,0,0), (0,1,0,0) to denote that one or more instances of R0 *OR* one or more instances of R1 can avoid deadlock). If not, provide a safe sequence.

| | **Yes** | **No** | **Additional Resource Required OR Safe Sequence** |
|---|---|---|---|
| In a deadlock? | | X | safe sequence is P0 followed by p1,p2, p3 in any order |

ii) It becomes Alice's turn and she ends up on a square where she must draw a mystery card. She reads her card: "You became R0-inefficient. You now need one more instance of R0." Therefore, Alice (P0) now needs (2,0,1,1) resources instead of (1,0,1,1). Alice wonders whether the existing allocation is now safe.

| | **Allocation** | | | | **Need** | | | | **Available** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R0 | R1 | R2 | R3 | R0 | R1 | R2 | R3 | R0 | R1 | R2 | R3 |
| P0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| P1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | | | | |
| P2 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | |
| P3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | | | | |

Mark an **X** in the appropriate box indicating whether the players are in a deadlock. If yes, list all additional resource vectors that could lead to avoid the deadlock state (e.g., list out (1,0,0,0), (0,1,0,0) to denote that one or more instances of R0 *OR* one or more instances of R1 can avoid deadlock). If not, provide a safe sequence.

| | **Yes** | **No** | **Additional Resource Required OR Safe Sequence** |
|---|---|---|---|
| In a deadlock? | X | | (1,0,0,0) or (0,0,1,0) |

**2) Memory Management [18 pts]**

a) **[12]** Eventually, Alice gets bored with OS Monopoly and decides to go back to plotting her revenge. Alice notices that Bob's toaster uses a variable-partition memory scheme. She checks the system, and finds that there are 5 memory partitions: A(300KB), B(400KB), C(200KB), D(700KB) and E(400KB). She tries to start up 4 processes (sequentially) that use 321KB, 356KB, 531KB, and 127KB. However, being the toaster it is, Alice once again cannot tell what's happening. Alice comes to you for help to simulate what happens with the first-fit, best-fit, and worst-fit memory management algorithms. Being the smart friend you are, you tell Alice which partition (letter) each of the processes is allocated to, and how many KBs are left in the partition after the process would start. However, if a process cannot be allocated, the system crashes, which you denote by writing "/" in all blanks after that point

| | 321KB | 356KB | 531KB | 127KB |
|---|---|---|---|---|
| First-fit (FF) | Partition: B<br>Leave 79 KB | Partition: D<br>Leave: 344 KB | Partition: /<br>Leave: / | Partition: /<br>Leave: / |
| Best-fit (BF) | Partition: B<br>Leave 79 KB | Partition: E<br>Leave 44 KB | Partition: D<br>Leave 169 KB | Partition: D<br>Leave 42 KB |
| Worst-fit (WF) | Partition: D<br>Leave 379 KB | Partition: B<br>Leave 44 KB | Partition: /<br>Leave / | Partition: /<br>Leave: / |

b) **[6]** Alice doesn't quite understand what you've written or done, but knows that for OS class, she must understand some things about FF, BF and WF. She decides to ask you a few questions about the FF, BF, and WF algorithms, given the above. You decide to provide some answers and explanations to Alice. Mark **X** in the appropriate box, and briefly explain.

| | FF | BF | WF | Explanation |
|---|---|---|---|---|
| Which algorithm makes the most efficient use of memory? | | X | | Since only the Best-fit can allocate all processes in the memory |
| Assuming all the algorithms have succeeded in fitting all the processes, which algorithm is the best in terms of execution speed? | X | | | First fit because it doesn't have to scan all partitions |
| In general, which algorithm is good for minimizing the effects of external fragments? | | | X | WF tries to leave the biggest hole in the system, not small unusable free space |

**3) More Memory Management [10 pts]**

Alice realizes that she really doesn't quite understand anything going on, so she decides to "debug" using a much simplified system. Therefore, Alice decides to go online shopping and buys a 32-bit computer without a CPU cache. Reading the manual, Alice learns that this means that every memory read operation will access the memory. Continuing on, she reads that memory is organized with pages and page tables. Fetching a word (4 bytes) from memory to the CPU register takes 2 microseconds, but with the TLB in memory management unit (MMU), memory operations may take less time. The TLB holds pairs of virtual pages and physical page frames: <virtual page, physical page frame>. Alice decides to run some experiments to learn more about the TLB.

**a) [5]** Suppose that in the first experiment, Alice observes a 25% TLB hit rate and the access time for the TLB was 300ns. Help Alice by calculating the effective access time below, showing all work.

| | |
|---|---|
| | us |

EAT = 0.75(TLB miss) + 0.25(TLB hit)
= 0.75(TLB access time + memory access time + memory access time)
+ 0.25(TLB access time + memory access time)
= 0.75(0.3us + 2us + 2us) + 0.25(0.3us + 2us)
= 0.75(4.3us) + 0.25(2.3us)
= 3.8 us

**b) [5]** In the next experiment, Alice observes a 80% TLB miss rate and the access time for the TLB was 900ns. In this case, what would be the effective access time? Show all work.

| | |
|---|---|
| | us |

EAT = 0.8(TLB miss) + 0.2(TLB hit)
= 0.8(TLB access time + memory access time + memory access time)
+ 0.2(TLB access time + memory access time)
= 0.8(0.9us + 2us + 2us) + 0.2(0.9us + 2us)
= 0.8(4.9us) + 0.2(2.9us)
= 4.5 us

**4) Paging [15 pts]**
In the last part of Alice's preparation for her revenge, she takes another one of Bob's toasters and decides to look at the paging scheme. She learns that the system has a 40-bit logical address, page size of 32K, and requires 4 bytes per page table entry.

**a) [5]** Alice needs to know how many pages there are in the logical address space in order to know what she can or cannot do. However, she doesn't quite know how to assemble all the information she needs together, so she makes two simplifying assumptions: (i) the system uses two-level paging; and (ii) each page table can fit completely in a single frame. Write down the number of pages as a power of 2, and show your work.

Pages: 2^25

Work:

We need 15 bits of offset for addressing inside a 32K page. We have 40 - 15 = 25 bits for addressing pages. So we will have 2^(40 - 15) = 2^25 pages

**b) [10]** In addition to knowing the above, she must also find out the breakup of logical address bits (in particular, the offset bits, page table index bits, and page directory index bits). However, even after a few hours of googling, Alice happens to not find anything. You decide to help, and learn that there are only 32k entries in the page directory. Now she is wondering how many bits needed for the page directory and how many entries are in the page table.

**[5 pts] Number of bits for 1st level Page table: _____15_____**
log2(32k) = log2(32*1024) = 15

**[5 pts] Number of entries in 2nd level Page table: _____1024_____**
2^15 = 32k. We need 15 bits for the 1st level Page table. We already know that we need 15 bits for offset. So, for the 2nd level page table, we need (40-15-15) = 10. And the number of entries is 2^10 = 1024