CSCE 3613 Operating Systems Homework #3, ver. 1.1

Process Synchronization and Deadlocks and Memory Management

Name: **Robert Johnson**

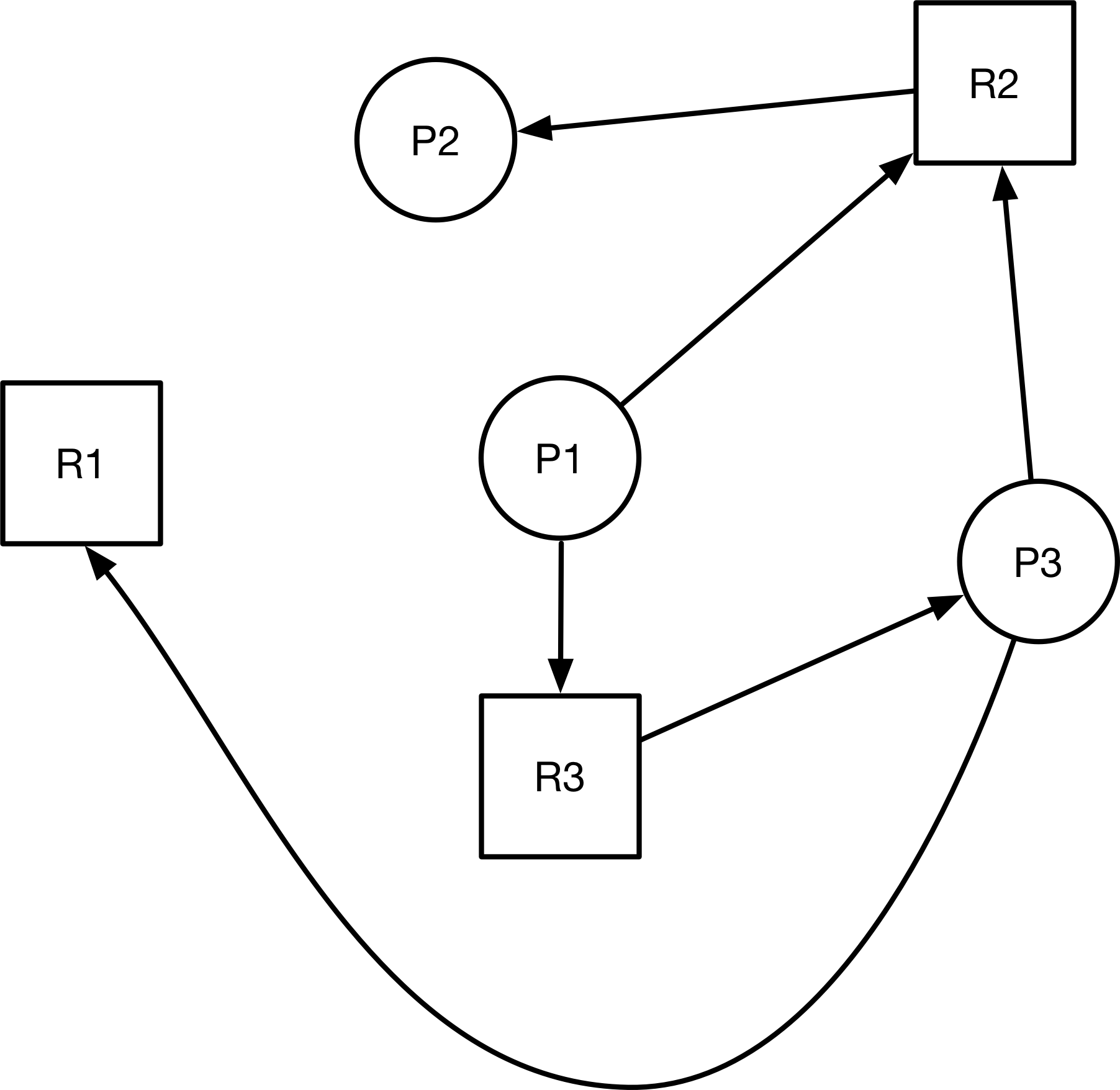
ID:  **010796992**

32 pts.

# **Instructions**

* Type your work, print it to a \*single\* PDF, and upload it to Blackboard before the due date and time. It is strongly suggested to use the given document.
* Show all of your work. Without proper justification and details of steps, correct answers alone may not carry full credit.
* -2 points if you do not insert your name and ID at the top of the document.
* -5 points if it is not typed.
* -5 points if it is not a PDF file.
* -5 points if it is not a single PDF file. Submit one PDF file. Do not submit zip files containing one or more files.
* -5 points if you present the worked problems out of order. In other words, please present the problems in the order assigned, 1, 2, 3, …

1. Consider the resource-allocation graph of a system below with 3 processes and 3 resources each with a single instance.



1.a. (5 pts.) Is the state of the system safe? Explain.

**Yes, the system is safe from deadlock because there is no cycle.**

1.b. (3 pts.) Starting with the original resource-allocation graph, assume R1 is assigned to P1. With this assignment, is the system safe or unsafe? Explain why.

**With this assignment, the system is now unsafe because it has created a cycle (P3 now requests R1 that is assigned to process P1 while P1 requests R3 that is assigned to P3) where the resource categories contain only a single instance therefore deadlock exists.**

1.c. (3 pts.) Starting with the original resource-allocation graph, assume R1 is assigned to P2. With this assignment, is the system safe or unsafe? Explain why.

**With this assignment, the system would still be safe because a cycle still wouldn’t exist.**

2. The following system has 11 hard drives and each process declares its maximum needs. In addition, the current needs (hard drives) that each process is holding is listed.

|  |  |  |
| --- | --- | --- |
| Process | Maximum Needs | Current Needs |
| P0 | 4 | 1 |
| P1 | 11 | 4 |
| P2 | 7 | 1 |
| P3 | 3 | 2 |
| P4 | 6 | 2 |

2.a. (5 pts.) What is a safe sequence for this system? Prove it.

**P3>P0>P4>P2>P1**

**Current needs only use 10 out of 11 hard drives so 1 hard drive is available. P3 goes first when only 1 hard drive is available, after done, 3 are available. Now P0 can go, after it's done, 4 available, so P4 can go. P4 finishes, 6 available so P2 goes. P2 is done, 7 available so P1 can go.**

3. Consider a paging system with the page table stored in memory.

a. (2 pts.) If a memory reference takes 250 nanoseconds, how long does a paged memory reference take? **500 nanoseconds.**

b. (2 pts.) If we add TLBs, and 80 percent of all page-table references are found in the TLBs, what is the effective memory reference time? Assume that finding a page-table entry in the TLBs takes 10 nanoseconds, if the entry is there. **10 + (.8\*250) + (.2\*500) = 310 nanoseconds**

4. Consider a logical address space of 32 pages with 1,024 words per page, mapped onto a physical memory of 16 frames.

a. (2 pts.) How many bits are required in the logical address?

**1024 = 2^10 (10 bits), 32 = 2^5 (5 bits) so, 10+5 = 15 bits**

b. (2 pts.) How many bits are required in the physical address?

**1024 = 2^10 (10 bits), 16 = 2^4 (4 bits) so, 10+4 = 14 bits**

5. Consider the two-dimensional array A:

int A[][] = new int[100][100];

where A[0][0] is at location 200 in a paged memory system with pages of size 200. In addition, in this system arrays are stored in memory in the following order: A[0][0], A[1][0], … A small process that manipulates the matrix resides in page 0 (locations 0 to 199). Thus, every instruction fetch will be from page 0.

For three page frames, how many page faults are generated by the following array-initialization loops, using LRU replacement and assuming that page frame 1 contains the process and the other two are initially empty?

a. (2 pts.)

for(int j=0; j<100; j++){

for(int i=0; i<100; i++){

A[i][j]=0;

}

}

**Answer : 100 \* (100/2) = 5000 page faults**

b. (2 pts.)

for(int i=0; i<100; i++){

for(int j=0; j<100; j++){

A[i][j]=0;

}

}

**Answer: 100 / 2 = 50 page faults**

6. A certain computer provides its users with a virtual memory space of 2^32 bytes. The computer has 2^18 bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456 hexadecimal.

a. (2 pts.) How many entries are there in the page table?

**2^32/2^12 = 2^20 = 1,048,576 entries**

b. (2 pts.) Explain how the system establishes the corresponding physical location.

**The system will take the virtual address and put it into binary form which gives us 32 bits. Then it looks at the page size and the page table size. In this example, the page size is 2^12 so the lowest 12 bits will be the offset. The page table size is 2^20 so the remaining 20 bits will be the index. It then finds the entry at whatever the index is to find the physical frame number. Finally, it combines the physical frame number with the offset to get the physical location.**