United International University
Department of Computer Science and Engineering
CSE 4509/CSI 309: Operating System Concepts/Operating Systems
Final Examination: Spring 2025
Total Marks: 40 Time: 2 hours

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

Answer all the questions. Numbers to the right of the questions denote their marks.

1 Suppose you are writing an operating system for a machine with 256 MB of RAM and 32 bits virtual address. To virtualize the memory, you have decided to use page tables with 1 KB pages. Due to scarcity of memory, you have to use paging. Page Table Entry (PTE) size for the page table is 4 bytes.

a) What is the size of the offset? [1]
b) In this system, what is the size of the physical address (in bits)? [2]
c) How much memory will a single level page table use? [2]
d) Design a multi level page table. [2]

2 Imagine a computer system with a physical memory that **can hold up to 4 pages.** This system uses virtual memory management with a page replacement policy based on the **Least Recently Used (LRU)** algorithm. You are provided with a sequence of page references and must simulate the behavior of the system as it processes this sequence. [5]

The sequence of page references is as follows:
0, 1, 2, 3, 0, 4, 2, 1, 0, 5, 2, 3, 4, 1

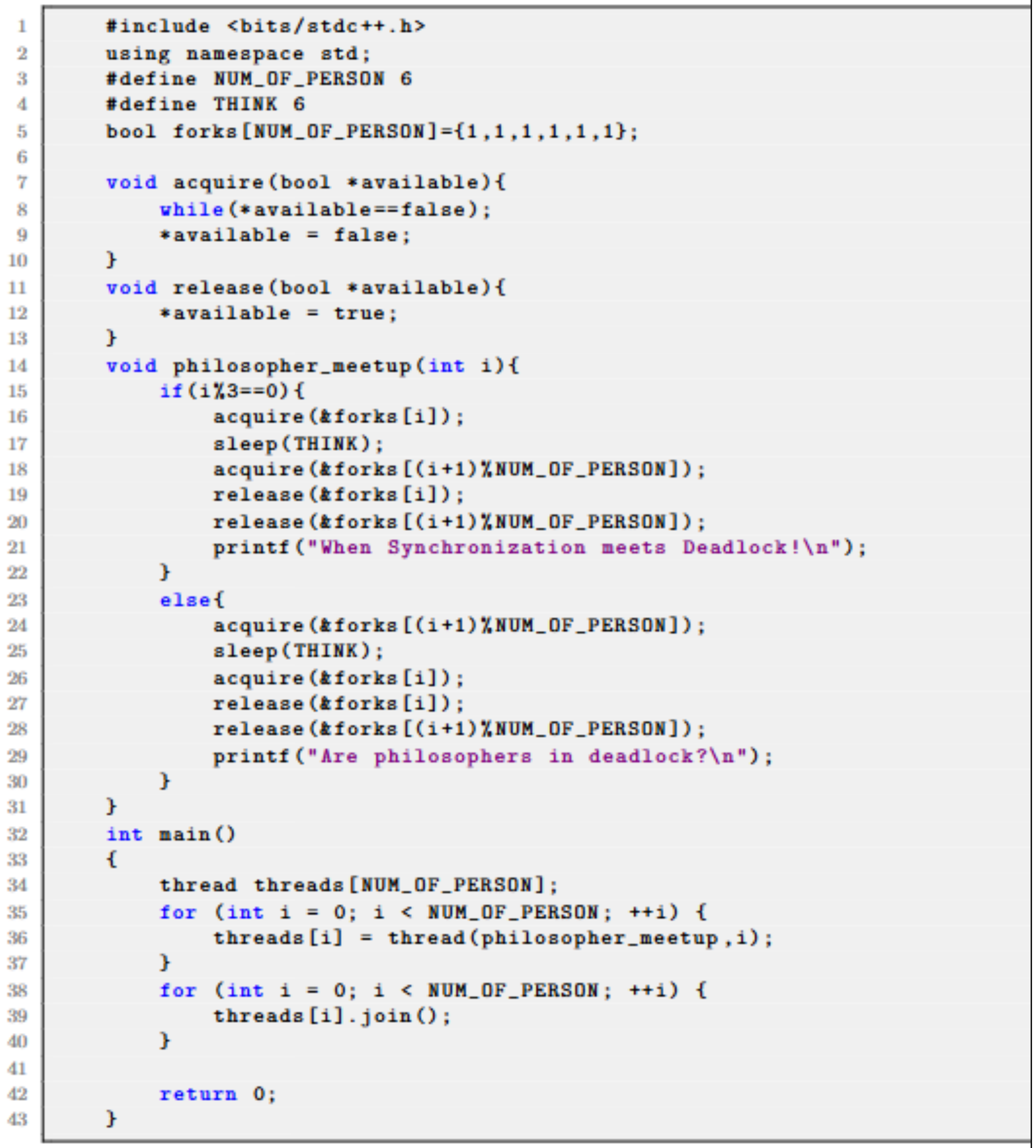
3 A system uses 16-bit virtual addresses with a page size of 256 bytes. The physical memory is 4 KB. [3]

VPN	0	1	2	3	4	5	6	7	8	9	10	…
PFN	3	15	20	6	1	9	7	11	17	8	4	…
Valid	1	1	0	1	1	1	0	1	0	0	1	…

Translate the following virtual addresses to physical addresses or report a page fault:

a) 0x02A3
b) 0x064B
c) 0x0A10

4 Look at the code snippet below and answer the questions.



```
 1  #include <bits/stdc++.h>
 2  using namespace std;
 3  #define NUM_OF_PERSON 6
 4  #define THINK 6
 5  bool forks[NUM_OF_PERSON]={1,1,1,1,1,1};
 6
 7  void acquire(bool *available){
 8      while(*available==false);
 9      *available = false;
10  }
11  void release(bool *available){
12      *available = true;
13  }
14  void philosopher_meetup(int i){
15      if(i%3==0){
16          acquire(&forks[i]);
17          sleep(THINK);
18          acquire(&forks[(i+1)%NUM_OF_PERSON]);
19          release(&forks[i]);
20          release(&forks[(i+1)%NUM_OF_PERSON]);
21          printf("When Synchronization meets Deadlock!\n");
22      }
23      else{
24          acquire(&forks[(i+1)%NUM_OF_PERSON]);
25          sleep(THINK);
26          acquire(&forks[i]);
27          release(&forks[i]);
28          release(&forks[(i+1)%NUM_OF_PERSON]);
29          printf("Are philosophers in deadlock?\n");
30      }
31  }
32  int main()
33  {
34      thread threads[NUM_OF_PERSON];
35      for (int i = 0; i < NUM_OF_PERSON; ++i) {
36          threads[i] = thread(philosopher_meetup,i);
37      }
38      for (int i = 0; i < NUM_OF_PERSON; ++i) {
39          threads[i].join();
40      }
41
42      return 0;
43  }
```

a) Draw the resource allocation graph for the above code. [3]

b) Is there a deadlock? If so, identify the philosopher IDs responsible for the deadlock. Otherwise, determine the output. [1]

c) What will happen if we switch the statement of lines 17 and 19? [2]

5 Consider the Table-1 snapshot of a system:

Process	Allocation				Max			
	A	B	C	D	A	B	C	D
T0	3	0	1	4	5	1	1	7
T1	2	2	1	0	3	2	1	1
T2	3	2	2	1	3	3	2	1
T3	0	5	1	0	4	6	1	2
T4	4	2	1	2	6	3	2	5

Table 1: Process along with resources

a) Determine whether the system is in a safe state or not if the available resources are {0,3,0,0}? If the state is safe, illustrate the order in which the threads may be completed. Otherwise,Calculate the minimum number of additional available resources required to ensure the execution of all threads. [4]

b) "If a request from thread T0 arrives for (0,0,1,2), the request will be granted immediately."- Justify the statement. [Assume, available resources={1,0,1,3}] [2]

6 a) What is a dangling pointer in the context of acyclic graph-based directory structures? Discuss how the issue of dangling pointers can be resolved. [3]

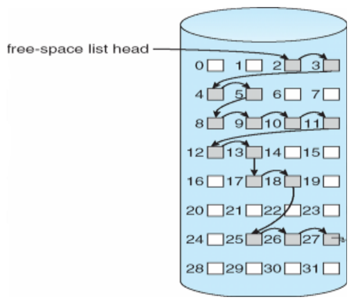
b) Consider the following directory organization:

```
Spring 2025
├── Computer Network
│   └── Routing Protocols.pptx
└── Operating System
    └── Deadlocks.pptx
```

Show this directory organization using Single Level Directory Structure. What problem do you find while implementing this? How does the Tree structure solve that problem? [3]

7 a) A file consists of 10 blocks and requires frequent random access to different portions of its content. Which file allocation method is most appropriate in this scenario, considering the need to avoid external fragmentation? Justify your answer. [3]

b) Consider the following diagram: [4]



Why is it inefficient to find contiguous free space using this method shown in the diagram? Can you suggest some modification(s) in this method to make it efficient?