1. (10 pts) **Deadlock.** Short answer questions:

   (a) A system has six tape drives \((a, b, c, d, e, f)\), with \(n\) threads competing for them. Each thread may need two of the drives. For what values of \(n\) is the system deadlock free?

   **Solution:** One thread. For two threads for example, we can get deadlock with the following:

   **Example**
   
   Thread 1:
   
   a. Wait();
   b. Wait();
   ...

   Thread 2:

   b. Wait();
   a. Wait();

   (b) Can a system be in a state that is neither deadlocked nor safe? If yes, give an example system.

   **Solution:** Yes. For example, given 3 units of resource \(A\), if thread 1 has 2 units of \(A\) and its maximum is 3, and thread 2 has 1 and its maximum is 2. This state is not a safe, but if neither thread ever requests an additional unit of \(A\), then it is not deadlocked.

2. (20 pts) **Deadlock** Problem 8.9 from the textbook.

   Using the terminology defined in class (also defined in Sec 7.6.2), we have

   (a) \(\sum_{i=1}^{n} Max_i < m + n\)
   
   (b) \(Max_i \geq 1\) for all \(i\)

   Also, \(Need_i = Max_i - Alloc_i\). Assume there exists a deadlock. Then:

   (c) \(\sum_{i=1}^{n} Alloc_i = m\)

   Using (a) we get: \(\sum Need_i + \sum Alloc_i = \sum Max_i < m + n\)

   Using (c) we get: \(\sum Need_i + m < m + n\)

   That is, \(\sum_{i=1}^{n} Need_i < n\)

   This implies that there exists a process \(P_i\) such that \(Need_i = 0\). Since \(Max_i \geq 1\), it follows that \(P_i\) has at least one resource it can release. Hence, the system cannot be in a deadlock state.
3. (10 pts) **Deadlock.** Consider the following system snapshot using the data structures in the Banker’s algorithm, with resources A, B, C, and D, and processes P₀ to P₄.

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Max</th>
<th>Available</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C D</td>
<td>A B C D</td>
<td>A B C D</td>
<td>A B C D</td>
</tr>
<tr>
<td>P₀ 3 0 0 2</td>
<td>6 0 1 2</td>
<td>3 0 1 0</td>
<td></td>
</tr>
<tr>
<td>P₁ 1 0 0 0</td>
<td>1 7 5 0</td>
<td>0 7 5 0</td>
<td></td>
</tr>
<tr>
<td>P₂ 1 3 5 4</td>
<td>2 3 5 6</td>
<td>1 0 0 2</td>
<td></td>
</tr>
<tr>
<td>P₃ 0 6 3 2</td>
<td>1 6 5 2</td>
<td>1 0 2 0</td>
<td></td>
</tr>
<tr>
<td>P₄ 0 0 1 4</td>
<td>1 6 5 6</td>
<td>1 6 4 2</td>
<td></td>
</tr>
</tbody>
</table>

Using Banker’s algorithm answer the following questions.

(a) How many resources of type A, B, C, and D are there?

**Solution:** \((\text{allocation} + \text{available}) = \{5,9,9,12\} + \{3,2,1,0\} = \{8,11,10,12\}\)

(b) What is the content of the \(\text{Need}\) matrix?

**Solution:** See above table.

(c) Is the system in a safe state? Why?

**Solution:** Yes, P₀ can finish with its current resources and what’s in available. When it finishes, avail becomes \(\{6,2,1,2\}\). Now, P₂ can complete and then avail would be: \(\{7,5,6,6\}\). Now, P₃ can complete and then avail would be: \(\{7,11,9,8\}\). Then either P₁ or P₄ can complete, followed by the other.

(d) If a request from process P₄ arrives for additional resources of \((1,2,0,0)\), can the Banker’s algorithm grant the request immediately? Show the new system state, and other criteria.

**Solution:** No the request cannot be granted because all of none of the process are able to request their max number of resources, i.e., for all processes \(i = 0, 4\), \(\text{need}(i) > \text{avail}(i)\).
4. (10 pts) Consider a segmented memory system with memory allocated as shown below.

Suppose the following actions occur:

- Process E starts and requests 300 memory units.
- Process A requests 400 more memory units.
- Process B exits.
- Process F starts and requests 800 memory units.
- Process C exits.
- Process G starts and requests 900 memory units.

(a) Describe the contents of memory after each action using the first-fit algorithm.

- E requests 300: E is allocated in 400-700
- A requests 400 more: cannot fit because the entire process is allocated in a single continuous chunk of memory in a segmented memory system. Need to compact memory: move B to 1100-1900, move E to 2400-2800, give A additional addresses 400-800
- B exits: there is a hole between 800-1600
- F requests 800: F is allocated in 800-1600
- C exits: there is a hole between 1600-2400
- G requests 900: no hole that is big enough. Compact memory: move E to 2800-3100, G is allocated in 1600-2500

(b) Describe the contents of memory after each action using the best-fit algorithm.

- E requests 300: E is allocated in 1600-1900
- A requests 400 more: this 400 is allocated in 400-800
- B exits: there is a hole between 800-1600
- F requests 800: F is allocated in 800-1600
- C exits: there is a hole between 1900-3100
- G requests 900: G is allocated in 1900-2800

(c) How would worst fit allocate memory?
- E requests 300: E is allocated in 2400-2700
- A requests 400 more: this additional 400 is allocated in 400-800
- B exits: there is a hole between 800-1900
- F requests 800: F is allocated in 800-16000
- C exits: there is a hole between 1600-2400
- G requests 900: no hole big enough. Need to compact: move E to 2800-3100, give 1600-2500 to G

(d) For this example, which algorithm is best?
For this example, best-fit is best.