1. A system using mailboxes has two IPC primitives, SEND(msg, destination) and RECEIVE(msg, source). The source parameter in the RECEIVE primitive specifies the process that must send the desired message, and blocks if no message from that process is available, even though messages may be waiting from other processes. There are no shared resources, but processes need to communicate frequently about other matters. Is deadlock possible? Explain.

2. Prove the time complexity of Bankers algorithm. (Derive the time, expressed as the number of operations using “big O” notation, required by Bankers algorithm to determine if a request for resources can be satisfied.)

3. Consider a system that runs 5000 jobs per month with no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about 10 jobs per deadlock. Each job is worth about 2 dollars (in CPU time), and the jobs terminated tend to be about half-done when they are aborted.

A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker’s algorithm) would be installed in the system with an increase in the average execution time per job of about 10 percent. Since the machine currently has 30-percent idle time, all 5000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average.

i) What are the arguments for installing the deadlock-avoidance algorithm?

ii) What are the arguments against installing the deadlock avoidance algorithm?

4. Consider a paged virtual memory system that has a page size of 512 bytes ($2^9$). Processes in this system can have a maximum virtual address space of 64K bytes ($2^{16}$). The system is currently configured with 5K bytes of physical memory.

a) How many frames in the physical address space?

b) How many pages in a virtual address space?
5. For a process $P$ (in the system described in question 4), the current state of physical memory is as follows:

<table>
<thead>
<tr>
<th>Address</th>
<th>Memory Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1536</td>
<td>Page 34</td>
</tr>
<tr>
<td>2048</td>
<td>Page 9</td>
</tr>
<tr>
<td>3072</td>
<td>Page Table</td>
</tr>
<tr>
<td>3584</td>
<td>Page 65</td>
</tr>
<tr>
<td>4608</td>
<td>Page 10</td>
</tr>
</tbody>
</table>

a) Show the current contents of the *entire* page table for process $P$.

b) What physical address is referenced by the virtual address: (Show your work!)
   - 4608
   - 5119
   - 5120
   - 33300
   - 33000

c) Show the contents of the page table after page 49 is loaded into frame 4.

6. Consider the two-dimensional array:

```plaintext
var A : array 1..100 of array 1..100 of integer
```

where $A[1][1]$ is at location 200, in a paged memory system with pages of size 200. A small process is in page 0 (locations 0 to 199) for manipulating the matrix; 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 page frame 1 has the process in it, and the other two are initially empty:

a) For $j := 1$ to 100
   For $i := 1..100$
     $A[i][j] := 0$
   End for
End for

b) For $i := 1$ to 100
   For $j := 1..100$
     $A[i][j] := 0$
   End for
End for
7. Assume the execution of a program generates the following address trace

\[ a b c d e e f d b a e e c \]

where \(a, b, c, d, e,\) and \(f\), are the pages that are referenced. For a memory system with 5 page frames (initially empty), show how the page replacement policies listed below would behave on the above address trace. Show the state of main memory at each step of the trace and show when page faults occur.

   a) FIFO Page Replacement
   b) LRU Page Replacement
   c) Working Set Page Replacement (assume a window size of 3 references)
   d) Why is it that the FIFO replacement policy produces the lowest number of page faults? (Is this surprising?)

8. Consider a computer with 32-bit addresses that uses a two level page table. Virtual addresses are split into a 9-bit top-level page table field, an 11-bit second-level page table field, and an offset. How large are the pages and how many are there in the address space?

9. Consider a system with memory mapping done on a page basis and using a single-level page table. Assume that the necessary page table is always in memory.

   a) If a memory reference takes 60ns, how long does a paged memory reference take?
   b) Now we add an MMU with a TLB that imposes an overhead of 5ns on a hit or a miss. If we assume that 85% of all memory references hit in the MMU TLB, what is the effective memory access time when the page is in memory?
   c) Now assume that the probability that a referenced page is in memory is 90% (i.e., 10% of all page references result in a page fault). Assume that TLB hits only occur for pages in memory. Assume it takes 20ms to service a page fault. What is the effective memory access time (using the MMU) when page faults are considered?

10. It has been observed that the number of instructions executed between page faults is directly proportional to the number of page frames allocated to a program. If the available memory is doubled, the mean interval between page faults is also doubled. Suppose that a normal instruction takes 1 microsecond, but if a page fault occurs, it takes 2001 microseconds (i.e., 2 ms to handle the fault). If a program takes 60 seconds to run, during which time it has 15,000 page faults, how long would it take to run if twice as much memory were available?