1. [10 points] Consider the following segment table:

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>219</td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>2300</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>1327</td>
<td>580</td>
</tr>
<tr>
<td>4</td>
<td>1952</td>
<td>96</td>
</tr>
</tbody>
</table>

What are the physical addresses for each of the following logical addresses?

- 1, 20
- 3, 450
- 6, 90

All values are in decimal.

2. [10 points] Consider the following page table. All numbers in the table are binary.

<table>
<thead>
<tr>
<th>Page Number</th>
<th>Frame Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1101</td>
</tr>
<tr>
<td>0001</td>
<td>0011</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
</tr>
<tr>
<td>0011</td>
<td>0000</td>
</tr>
<tr>
<td>0100</td>
<td>1101</td>
</tr>
</tbody>
</table>

- Given the physical address of each of the following logical addresses. Assume that addresses occupy 12 bits.
  - 001100000010₂ (base 2)
  - 400₁₀ (base 10)
  - 000001100111₂ (base 2)
- What is the page size in this system?

3. [15 points] Suppose we have a page trace \{4,3,2,1,4,3,5,4,3,2,1,5\}. Use FIFO, LRU and MIN (the optimal algorithm) to run this page trace with 3 page frames and then with 4 page frames, and report the following:

- For each page replacement algorithm, use two tables to show the page replacement activities, one table for 3 page frames and the other for 4 page frames.
- For each table, clearly show the number of page faults, hit ratio, and miss ratio.
- For each page replacement algorithm, indicate Belady anomaly if there is any.
- In each table, circle the pages that were brought into physical memory due to page faults.
• The memory content may or may not change after each page reference. Compare the 3-frame table and the 4-frame page, verify that after each page reference whether the 4-frame column contains the 3-frame column in terms of the pages on that column. This is to verify the “inclusion property”. For each column, circle the page in the 4-frame table that is not in the 3-frame table.

Note that near the end of the MIN (optimal) algorithm, there could be multiple choice of pages to be evicted. To ensure we will speak the same language, if you have a multiple choice, always select the page with the smallest number to evict. Use the table format used in our class slides.

4. [10 points] As discussed in class that we could reorder the pages on a column according to how recent a page is referred to. More precisely, the most recently referenced page is at the top, while the least recently referenced page is at the bottom. Between the top and bottom, the pages are ordered in a similar way. Suppose the column currently has pages 3 (top), 2, 4, 1, 0 (bottom) in this order. If the next page is page 1, which becomes the most recently used page, pages on that column are re-ordered as 1 (top), 3, 2, 4, 0 (bottom). If the next reference is page 5, which is not in physical memory, then it is brought into memory and the pages on that column becomes 5 (top), 1, 3, 2, 4, 0 (bottom). What if we only have 5 page frames? Simple, the bottom page is removed and the result is 5 (top), 1, 3, 2, 4 (bottom).

Problem (3) has a page trace of 12 page references. Run that page trace with 3 page frames, 4 page frame and 5 page frames. However, pages on each column in the table must be reordered in the indicated way so that the most (resp., least) recently used page appears at the top (resp., bottom). Then, carefully examine the corresponding table columns, you should quickly get a sense (1) that if a page fault occurs, a newly referenced page is pushed to the top while the other pages are pushed downward, and the least recently used page is popped out from the bottom; and (2) that if there no page fault, a page somewhere on the column is taken out, pushed into the top position. and all pages between the top and the position of the page being referenced are pushed down. Isn’t this a stack operation? Therefore, LRU is a stack algorithm, and each column is a stack. For the optimal MIN algorithm, the stack is ordered based on how soon a page will be used. At the top is a page that will be used the shortest period of time, and the bottom is a page that will not be used for the longest period of time. Therefore, the optimal MIN algorithm is also a stack algorithm.

Do the following problems:

• Trace the page trace using LRU and MIN with 3 frames, 4 frames and 5 frames.
• Combined the 3-frame, 4-frame and 5-frame into a single table of 5 rows such that the first 3 rows correspond to the 3-frame results, the first 4 rows correspond to the 4-frame results, and the first 5 rows correspond to the 5-frame results. Is this doable?
• If it can be done, what you can find from this? Present a detailed discussion.

This problem is related to Problem (6).

5. [10 points] A process has four page frames allocated to it. The table below includes the following items: (1) the time of the last loading of a page into each frame, (2) the time of last access to a page in each frame, (3) the virtual page number in each frame, and (4) the reference (R) and modified (M) bits for each frame. The times are in clock ticks, from the process starts at time zero to the event.

<table>
<thead>
<tr>
<th>Virtual Page No.</th>
<th>Page Frame</th>
<th>Time Loaded</th>
<th>Time Used</th>
<th>R-bit</th>
<th>M-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>60</td>
<td>150</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>130</td>
<td>160</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>70</td>
<td>155</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100</td>
<td>145</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A page fault to virtual page 4 has occurred. Which page frame will have its content replaced for each of the following page replacement algorithm: FIFO, LRU and Clock? Explain why in each case.

6. [10 points] Let \( P = \langle p_1, p_2, \ldots, p_n \rangle \) be a page trace of size \( n \) and let \( m \) be the number of page frames. Also, let \( M_t(P, \alpha, m) \) be the memory content (i.e., pages) after referring to page \( p_t \) with respect to page replacement algorithm \( \alpha \). A page replacement algorithm satisfies the Inclusion Property if \( M_t(P, \alpha, m) \subseteq M_t(P, \alpha, m + 1) \) for every \( t \). This expression suggests that the content in memory \( M_t(P, \alpha, m) \) after referring to page \( p_t \) is a subset of \( M_t(P, \alpha, m + 1) \) that has one more page frame. In other words, for every page reference \( p_t \), the content in memory with \( m \) page frames is a subset of the content in memory with \( m + 1 \) page frames.

Do the following problems:

- Prove that if a page replacement algorithm satisfies the inclusion property, then Belady anomaly cannot happen!
- Prove that the LRU algorithm does satisfy the inclusion property. (Hint: Recall the way of ordering the pages on a column based on how long a page was not used. (See Problem 4.) The most recently used page is at the top, while the least recently used page is at the bottom. When a page fault occurs, the bottom-most page is evicted (if there is no free page frames). Now, what would happen when each column has one more page frame! This is not a difficult problem if you understand the concept of “stack algorithms”.

If you are above to prove LRU satisfies the inclusion property, you should also be able to prove the optical algorithm MIN also satisfies the inclusion property.

7. [10 points] The working set (WS) of a process at virtual time \( t \), written as \( W(t, \theta) \), is the set of pages that were referenced in the interval \((t - \theta, t]\), where \( \theta \) is the window size.\(^1\) The problem with working set is that it is rather difficult to implement accurately. In 1972, Chu and Opderbeck proposed an interesting alternative to working set.\(^2\) This is the Page Fault Frequency (PFF) replacement algorithm. Unlike WS which updates \( W(t, \theta) \) after every page reference, PFF suggests that the set be updated only when a page fault occurs. More precisely, let \( t \) and \( t' \) be two consecutive page fault times, where \( t' < t \). Then, the page that should be in physical memory at time \( t \) and window size \( \theta \) \( P(t, \theta) \) is defined as follows:

\[
P(t, \theta) = \begin{cases} 
W(t, t - t' + 1) & \text{if } t - t' > \theta \\
P(t', \theta) \cup \{p_t\} & \text{otherwise}
\end{cases}
\]

This means the following:

- If two consecutive page faults have a longer time span than the window size \( \theta \), then \( P(t, \theta) \) is the working set from time \( t' \) \( t + 1 \).
- Otherwise, \( P(t, \theta) \) is the older \( P(t', \theta) \) plus the newly referenced page \( p_t \).

The following is a table showing the PFF set for page trace \{4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5\} and \( \theta = 2 \).

---


In this table, from $t = 1$ to $t = 4$ we have four page faults, and the time gap between two consecutive page faults is 1. Because these gaps are less than $\theta = 2$, $P(t, \theta)$ increases after every page fault. Note that the way of presenting this table resemble a stack on each column. On each column the order of the page numbers is based on the distance from now and the last reference. In this way, the most recently referenced page is at the top of that column (i.e., stack). When a page is referenced, pages on that column are reordered. For example, after referring to page 1 at $t = 4$, the column has $\{1, 2, 3, 4\}$. The next page is 4, which is at the bottom previously, moves to the top of the column for $t = 5$.

There is no page fault at time $t = 5$ and $t = 6$ and $P(5, \theta)$ and $P(6, \theta)$ do not change; but, page ordering is changed. At time $t = 7$ page $p_7 = 5$ causes a page fault. Because the last page fault was $t' = 4$ and the gap $t - t' = 7 - 4 = 3 > \theta = 2$, $P(7, \theta)$ should include all pages referenced between $t'$ and $t$. Therefore, $P(t, \theta) = \{5, 3, 4, 1\}$.

In the working set model, the number of pages in a working set is always no more than $\theta$; however, in the PFF model the number of pages in $P(t, \theta)$ can be larger than $\theta$ as shown in the above table. Study the PFF algorithm, use PFF and WS to process the page trace with $\theta = 3$:

\[
\{1, 2, 3, 4, 5, 4, 3, 5, 4, 3, 6, 7, 8, 1, 2, 3, 4, 5, 4, 3, 5, 4, 3, 2, 1, 6, 7, 7, 7, 8\}
\]

Now do the following:

- Use working set to process the above page trace with $\theta = 3$.
- Use PFF (Page Fault Frequency) algorithm to process the above page trace with $\theta = 3$.
- Draw a diagram with x-axis as the time (i.e., 1, 2, 3, ...) and the y-axis as the number of pages in physical memory.