The following questions are related to memory and virtual memory management. (Please provide “short” and “precise” answers to each sub-problem.)

1. External fragmentation will happen with segmentation, while internal fragmentation will occur with paging. However, it is also true that, when implementing a real OS using segmentation, we will still get internal fragmentation. Please explain why internal fragmentation will sometimes occur in segmentation. (*Hint:* Assume the size of the segment table is fixed and limited.)

2. Suppose we decide to use a paging scheme. We will then have virtual memory and, certainly, a good page-replacement algorithm is desired. From the Silberschatz & Galvin textbook, we know that LRU performs better than FIFO. *But, for one reason,* we cannot implement an *exact* LRU algorithm. We can only implement approximate LRU algorithms, like the second-chance algorithm. Please tell what this reason is. That is, tell why should we want to design LRU approximation algorithms instead of implementing the exact LRU.

3. Finally, we need to decide the page size. *If* we know that page faults will become more and more expensive and memory chips will cost less, *then* should we increase or decrease the page size? Please briefly explain the reason.
This question concerns semaphores and monitors. We will consider the implementation of each in terms of the other.

(a) (30 points) Implement a general semaphore in terms of a monitor. The P and V (sometimes called wait and signal) operations are defined as—

\[ P(s) : \text{wait until } s > 0; \text{then decrement } s \text{ by 1.} \]
\[ V(s) : \text{increment } s \text{ by 1.} \]

Code the \( P \) and \( V \) operations. Do not forget initialization!

(b) (20 points) Let a bounded semaphore \( s \) be a general semaphore that cannot exceed a given value \( smax > 0 \). The corresponding \( P \) and \( V \) operations are then defined as—

\[ P(s) : \text{wait until } s > 0; \text{then decrement } s \text{ by 1.} \]
\[ V(s) : \text{wait until } s < smax; \text{then increment } s \text{ by 1.} \]

Modify your design of part (a) to implement bounded semaphores.

(c) (20 points) In Section 6.7 of Silberschatz and Galvin, an implementation of a monitor is given in terms of semaphores. This implementation assumes that when a process signals another process, it goes into a different queue. Suppose that signaling processes wait along with new processes in the entry queue. Show how the code would be simplified. Give (i) the entry and exit code that surrounds each procedure of the monitor, (ii) the code for \( \text{wait} \) and (iii) the code for \( \text{signal} \).

(d) (20 points) Suppose that the use of \( \text{signal} \) is restricted so that it may occur only as the last instruction of any monitor procedure. Show how the code from Silberschatz and Galvin (possibly all three sections) would be simplified.

(e) (10 points) How could the code (all three sections) be simplified if the monitor contained no \( \text{wait} \) or \( \text{signal} \) operations?
1. [Run-time implementation model]

In Fortran it is possible to pass the label of a statement as a parameter to a function, and it is possible to return from that function by invoking the \texttt{goto} statement on that label. We would now like to provide such a facility in a block-structured, statically-scoped language such as Pascal or Ada. Picking Pascal as the example language we would have to change the syntax such that we can now declare label value parameters (in function definitions) and permit these labels to be targets of \texttt{goto} statements. For example, a procedure heading and the use of a label could be as follows:

\begin{verbatim}
procedure F(x: integer, y: label);
begin
    ... goto y;
end
\end{verbatim}

Discuss an implementation model to implement labels as parameters. Discuss specifically the code that should be generated for procedure calls and \texttt{goto} statements.
2. (a) Write a regular expression that captures comments of Pascal, namely a string of characters between (* and *). Explain your notation.

(b) Consider the statement

    if b do S1 else S2

Discuss the issues in generating code for such a statement in the context of a single-pass compiler.