Sean Sponsler
CSC453 Winter 2024

(Generally unsure of how much information you wanted here, started the report too late)

For this program, specific design choices I made not outlined in the book specification or modified specification include

- Using Python so I don't lose my mind with segmentation faults during extensive debugging, I am also very comfortable with Python
- Using a dictionary for page table, containing page numbers as keys and frame numbers (initialized to -1), loaded bit (initialized to False) as values for easier lookups
- Simply seeking the physical store with seek() to retrieve the value for a given page number
- Using the queue implemented in the FIFO algorithm generally to detect the size of my page table to determine if we need to replace a page, even if we are using LRU or optimal page replacement

My program procedure is as follows:

1. Initialize all variables, counters, constants necessary
2. Obtain user parameters, use defaults in given scenarios
3. Traverse through the referenced file, for each successive entry, translate the logical address, retrieve the page number and offset
4. Check the TLB if the page number is present, if not, increment TLB miss counter
5. On TLB miss, search page table for page number
    a. if found but not loaded, increment page fault counter (soft miss)
    b. if not found, increment page fault counter and determine if the page table is full, if so, call the relevant algorithm function according to the parameter
6. Once frame number is determined, load data from backing store and set loaded bit to true
7. Append page number to TLB
    a. If TLB is full at size 16, pop the front and append the new page number (FIFO)
8. Calculate the physical address and print associated info to console
9. Repeat (1-8) until file is completely read
10. Display statistics