Exercise 1 (1/3)

- In this exercise, You need to extend the implementation of ex2 from lab 08.
- You have to implement three functions as follows:

```
// Random page replacement
int random(struct PTE* page_table);
// NFU page replacement
int nfu(struct PTE* page_table);
// Aging page replacement
int aging(struct PTE* page_table);
```

Note: You can add more arguments to the functions if needed but all of them should have the same signature. The return value of the functions should be the page number to evict which is selected based on the algorithm, e.g. **nfu** function should select the page to evict based on the NFU page replacement algorithm.

Exercise 1 (2/3)

- The program **pager.c** should accept the page replacement algorithm as a command line argument (in addition to the number of frames and pages). The user can pass "random" for Random page replacement algorithm, "nfu" for NFU page replacement algorithm, or "aging" for Aging page replacement algorithm.
- The program **pager.c** should print an info message to the user about the selected algorithm.
- Extend the source code **mmu.c** to include a benchmarking functionality. The program **mmu.c** should calculate and print to stdout the *hit ratio* after finishing all requests considering that hit occurs when MMU gets a request to a valid page whereas miss occurs when it gets a request to an invalid page.

Exercise 1 (3/3)

- Run the program three times, one time for each algorithm using the same input. Compare the performance of the algorithms and add your findings to **ex1.txt**.
- A sample input for the program can be found in this gist.
- Submit **mmu.c**, **pager.c** and **ex1.txt** and a script **ex1.sh** to run your programs.
- **Note:** For Aging algorithms, assume that:
  - there is a timer interrupt for each page access request.
  - the counters are unsigned chars (8-bit integers) and each counter has n bits where $n = sizeof(\text{unsigned char})$. For NFU, assume that the the counter of type integer.
- To shift the variable num to the left x bits, we can write: `num << x` which equals to $num = num * (2^x)$.
- To shift the variable num to the right x bits, we can write: `num >> x` which equals to $num = num/(2^x)$.
- You can add referenced bit (r_bit) to the counter (8 bits) as follows: `counter = (counter >> 1) | (r_bit << 7)`

Exercise 2

- In this exercise, You need to extend the implementation of ex2 from lab 08. You have to add a TLB functionality to **mmu.c** program. The TLB here is represented as an array of **struct TLB_entry** which has only two fields as follows:

  ```
  struct TLB_entry {int page, frame;};
  ```

- In TLB, you need to store the most recent mappings between pages and frames. If you cannot find the requested page in TLB, then there is a TLB miss and you need to check the page table.
- We assume that the size of TLB is 20% size of the page table.
- Your program should calculate the TLB miss ratio and print it to stdout.
- Try different values for number of frames and pages. Compare the results and add the findings to **ex2.txt**.
- Submit **mmu.c**, **pager.c** and **ex2.txt**.
- Submit also a script **ex2.sh** to run your programs.