HW-8 CS451 10 points Due: Due: Monday 10/26/2020

The Clock page replacement requires 1 bit (the use bit)

associated with each page. Whenever we load or access a

page we set the use bit to 1. Whenever we want to replace

a page, we scan the buffer (implemented as a circular

linked-list) looking for a page with a use(u) bit set to 0.

If we find a page with a use bit of 1, we reset that bit

to 0 and move on.

The Not-Recently-Used (NRU) page replacement requires 2

additional bits associated with each page (a referenced

bit “R” and modified bit “M”). Whenever we read from or

write to a page we set the R bit to 1. Whenever we write

to a page we set the M bit to 1. When a process is started,

both bits for all its pages are set to 0. Periodically,

the OS resets the R bit to 0. When a page fault occurs

the OS inspects all pages and assigns them to 1 of 4

categories:

Class 0: R = 0, M = 0

Class 1: R = 0, M = 1

Class 2: R = 1, M = 0

Class 3: R = 1, M = 1

The OS removes (at random or first found) a page from the

lowest non-empty class set.

Your assignment is to implement a combined Not-Recently-Used & Clock page replacement algorithm that uses 2 bits (M and R) and a circular linked list where each page is represented by a node in the linked list. We will

call this the "Enhanced Second Chance - Clock"

algorithm (ESC-C).

To implement ESC-C, you must modify hw7 as follows:

The circular linked list will have 6 entries (can manage 6

pages) and will be managed by Morticia. Each of the 5 monster

threads will request at least 1 memory page when they are

created. If the account balance for any thread

is negative (after ANY "W") the thread will set the R and M

bits to 1 for that initially loaded page. If the account

balance for any thread is positive (after ANY "W") the thread

will set the R bit to 1 for that initially loaded page and

leave the M bit as it was. The ESC-C algorithm must reset

all R bits to 0 on occasion (I leave this to you to decide).

On occasion (e.g. randomly) each thread will require an

additional page, generating a page fault when all the pages have been filled. At this point the thread generating the page fault must:

1) print "Page fault in thread XXX",

2) locate a page to replace and print the details (R and M bits and ownership) of the page being removed,

3) load the new page, and 4) set the bits to some initial value (see

notes).

Once more than 1 additional page has been loaded, it is very

likely that a thread will have no pages in memory. That

thread will also generate a page fault whenever it encounters

a "W".

You will also need a mutex to protect the links in the linked

list to prevent any possible corruptions of the linked list.

NOTES:

------

Initially loaded pag