CS 3423 Operating Systems

Fall Semester 2019

Prof. Pai H. Chou

Assignment 11

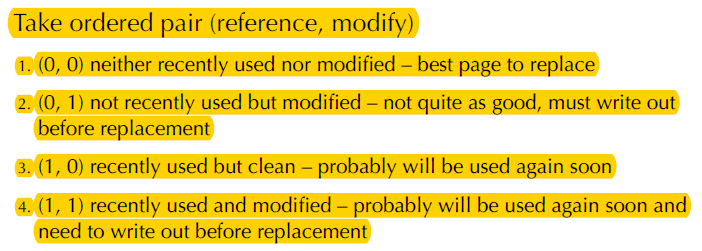
Due Date: Sunday, December 1, 2019, 11:59pm

Scope: Chapter 10, Virtual Memory

* + simplevm.py, typescript

## 1. Definitions and Short Answers

1. Of the several benefits of virtual memory, why is partial program loading a good idea?
   1. how does it improve performance (run faser)Less I/O needed to load or swap programs into memory
   2. how does it use less physical memory? because partial loaded
   3. how does it allow more processes to run at the same time? Each program takes less memory while running
2. Another benefit of virtual memory is that it allows the logical address space to be larger than physical. Why is this a good idea? program no longer constrained by limits of physical memory
3. What is the difference between **paging** and **swapping**? Swapping refers to copying the entire process address space; paging refers to copying in/out one or more pages of the address space
4. What does **demand paging** mean? bring page into memory only when needed What are its benefits compared to non-demand-paging? Less I/O needed, no unnecessary I/O, Less memory needed, Faster response, More users
5. When the **valid-invalid bit** in the page table entry is 'v', what does it mean for demand paging? valid, page in memory
6. When the valid-invalid bit is 'i' in the page table entry, what are two possibilities? How does the OS handle them? page not in memory or invalid reference
7. What does **zero-fill-on-demand** mean? upon allocation, initialize entire page to 0 When is it used and why? privacy protection - don't want previous process's data to be seen by another process
8. Of the costs of page fault, which has the highest cost?
   1. servicing the interrpt
   2. restart the process
   3. page-in, page-out Most time spent
9. What is **copy-on-write** and how does it help make fork() run more efficiently?Allows both parent and child processes to initially share the same pages in memory when fork()
10. Does vfork() use copy on write?NO How does it work for a shell to launch a process? child changes will be visible to parent, sharing stops when exec() is called.
11. What does a **frame allocation algorithm** decide? How many \_\_frames to allocate to \_\_\_process?
12. What does a **page replacement algorithm** decide? Which \_\_\_frame to replace when \_\_\_no free frame available?
13. What is the Belady's Anomaly? It means you could get more \_\_\_page fault when you have more \_\_\_frame? It is common in \_\_\_FIFO based algorithms.
14. What is the Optimal page replacement algorithm? It is to replace the page that \_\_\_\_\_\_\_will not be used for longest period of time ? Can it be implemented exactly? NO
15. What is a good page replacement algorithm in practice? LRU
16. To implement LRU,
    1. What does a Counter store time, when is it updated every memory reference, and how is it used? search table to find smallest value
    2. how does a stack algorithm work when the page being referenced is already on the stack? move to top
17. Of LRU approximation algorithms,
    1. In **single reference bit algorithm**, what page gets replaced? bit = 0
    2. in **additional reference bits algorithm**, how are multiple reference bits for each page maintained? 每次reference都向右移, 被reference到的就第1個bit+1
    3. How does **second-chance algorithm** decide what page to replace? reference bit = 0 What update to reference bit does it do under what condition? reference bit = 1 -> 設成0，循環check
    4. in **enhanced second chance** algorithm, what are the four combinations in order of replacement priority? (reference bit, modify bit)



1. In Counting-based algorithms,
   1. what is the issue with least-frequently used (LFU) algorithm? large count can't get replaced easily
   2. What is the rationale behind most frequently used (MFU) algorithm? page with the smallest count 可能只是剛拉進來，之後還要再用
2. In Page-buffering,
   1. How does it make sure there are always free frames available by the time a page fault occurs? 逐出victim的時機不在一定要選victim的時候，有空就踢人
   2. An extended version keeps a list of modified pages. What is the purpose? When backing store idle, write pages there and set to non-dirty
   3. Another extended version is to retain the frame contents even when the frame is put on the free frame list. What is the purpose? reduce penalty if wrong victim frame selected, referenced again before reused, no need to load contents again from disk
3. What are two schemes for **fixed allocation** of frames to processes? equal allocation, proportional allocation
4. In **priority allocation** of frames, upon page fault, how would OS **select victims** to reflect priority of processes?select victim from a process with lower priority
5. Between **global replacement** and **local replacement**,
   1. What are the advantages and disadvantages of global replacement?

adv: greater throughput and utilization

disadv: less predictable execution time

* 1. What are the advantages and disadvantages of local replacement?

adv: more consistent per process performance

disadv: underutilized memory

1. What is the difference between a **major page fault** page not in memory and a **minor page fault**? page in memory but not be mapped
2. What is the purpose of a **reaper**? 控制free page的數量
3. What is the meaning of **thrashing**? process忙於swapping page in and out What causes thrashing? 低的CPU utilization -> 增加更多process -> 更多thrashing
4. How does the **working set model** approximate locality?total demand frames And what condition implies thrashing? total demand frames > available frames
5. in the page fault frequency (PFF) strategy, what does it mean acceptable page-fault frequency when the page fault rate is too high and what does the OS do? OS add frame to process Too low?OS take frame from process
6. What is different between physical memory allocation **for kernel** vs. for user process? for kernel: requests memory for structures of varying sizes ; for user: logically contiguous, but not physically
7. How does the buddy system decide how much memory to allocate to serve a request? using power-of-2 allocator What are its advantages and disadvantages?

adv: 快速合併unsed chunk

disadv: fragmentation(internal ?)

1. In SLAB allocation,
   1. What is a slab one or more physically contiguous pages and how big does it have to be可以contain一個或多個instances ?? What does a slab contain? given type of kernel data structure
   2. How many **caches** are there? one or more slabs What does a cache contain? unique type of kernel data structure
   3. Why is there **no fragmentation** in slab scheme? 最小單位是object不是page或是chunk

## 2. Programming Exercise

In this programming exercise, you are to implement paging algorithms for OPT, FIFO, LRU, and SecondChance in Python. To do this, create a SimpleVM class using the following [template](https://drive.google.com/file/d/13-woRgRBegJUVea5aUULqb8ePhytb1b3/view?usp=sharing) and rename it **simplevm.py**.

class SimpleVM:

\_ReplacementPolicies = ['OPT', 'LRU', 'FIFO', 'SecondChance']

def \_\_init\_\_(self, numPages, numFrames, replacementPolicy):

self.numPages = numPages

self.numFrames = numFrames

if not replacementPolicy in SimpleVM.\_ReplacementPolicies:

raise ValueError('Unknown replacement policy %s' % replacementPolicy)

self.replacementPolicy = replacementPolicy

self.pageTable = [None for i in range(numPages)]

self.valid = ['i' for i in range(numPages)]

self.frames = [None for i in range(numFrames)] # storage

self.dirty = [False for i in range(numFrames)]

# we prefill swapspace content with chars '0','1','2'..  
 self.swapSpace = [chr(ord('0')+i) for i in range(numPages)]  
 # a frameTable maps a frame to the page, if any.

self.frameTable = [None for i in range(numFrames)]

# policy-specific code here

if self.replacementPolicy == 'LRU':

# use a "stack" (really more like a queue) to track age.

self.stack = []

if self.replacementPolicy in ['FIFO', 'SecondChance']:

# both FIFO and SecondChance are somewhat like RR

# so you could either use a circular buffer or perhaps  
 # keep index. Your own code here!

if self.replacementPolicy == 'SecondChance':

self.reference = [False for i in range(numFrames)]

def getFreeFrame(self, pageNum):

# find a free frame if any, or return None if not found.  
 # see comment in template for more info.

for i in range(self.numFrames):

if self.frames[i] is None:

return i

return None

def pickVictim(self, future=None):

# finds a page whose frame is to be evicted to fulfill page fault.

# this is called only if getFreeFrame returns None

if self.replacementPolicy == 'OPT':

# use future knowledge to pick victim

if future is None:

raise ValueError('cannot pick OPT without future')

# Your code here!!!

# find page that won't be used for longest time in future

# Note if future is empty list, then any page is ok!

# in any case, return the victim page's frame number.

if self.replacementPolicy == 'LRU':

# Your code here!! pull the victim from the bottom of the stack

# the assumption is if we have free frame in the first place,

# we would not need to evict anybody.

if self.replacementPolicy == 'FIFO':

# Your code here!  
 # pick victim in FIFO order

if self.replacementPolicy == 'SecondChance':

# Your code here!!

# base on referenceBit

# if we have not returned by then, it is an unknown policy

raise ValueError('unknown poliy %s' % self.replacementPolicy)

def pageIn(self, frameNum, pageNum):

# Your code here!!  
 # called to bring in a page from swap space to the frame.

# pageNum is used to find location in swap space.

# for simplicity, we use pageNum to index into swap space.

# Assume frameNum is free, and thus no page is currently using it.

# Update the page-to-frame table and frame-to-page table,

# set valid bit for the page, and clear dirty bit for the frame.

# in case of SecondChance, also clear reference bit.

def pageOut(self, frameNum):

# Your code here!!

# this flushes a frame (for a given pageNum) to swap space.

# Note that we only mark it as not-dirty, but it does not

# change state of valid bit because that is someone else's decision

# whether they want to reclaim the page or just flush it.

# Similar to pageIn, we assume swap space uses the virtual address

# as we have only one process.

def getFrame(self, pageNum, future=None):

# this is a utility that may be helpful, but not required.

# - see if pageHit, if so, return valid frame # for read/write.

# - if pageFault,

# - see if free frame available; if so, grab it;

# - but if no free frame, pick victim, page out first,

# fall thru to page-in

# - page-in and return the frame number

# - bookkeeping: look up the page# whose frame will be reassigned

# - set its pageTable entry to None, clear that page's valid bit

# finally, return the frame number for caller to use.

def updateAccess(self, frameNum, write=False):

# Your own code!! in different cases below!

if self.replacementPolicy == 'LRU':

# Your code here!! find frame in stack; if found, pop it.

# in either case, push back on stack.

if self.replacementPolicy == 'SecondChance':

# Your own code!! - mark the reference bit

if write: # for future use, if supporting write-access

self.dirty[frameNum] = True

def readPage(self, pageNum, future=None):

# Your code here!!

# get the frame number -- can call the getFrame() method for this.

# use the frame number to get the data so we can return it.

# do some bookkeeping by calling updateAccess

def writePage(self, pageNum, data, future=None):

# Your code here!!

# analogous to the readPage, except

# the frame is written to with data.

# do bookkeeping with write=True

You will find the test cases in the template file.

Here is a sample output of the program:

$ python3 simplevm.py

-------------- policy (read): OPT--------------

readPage(7)='7', pageTable=[None, None, None, None, None, None, None, 0], valid=iiiiiiiv, frames=['7', None, None]

readPage(0)='0', pageTable=[1, None, None, None, None, None, None, 0], valid=viiiiiiv, frames=['7', '0', None]

readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(4)='4', pageTable=[None, None, 0, 2, 1, None, None, None], valid=iivvviii, frames=['2', '4', '3']

readPage(2)='2', pageTable=[None, None, 0, 2, 1, None, None, None], valid=iivvviii, frames=['2', '4', '3']

readPage(3)='3', pageTable=[None, None, 0, 2, 1, None, None, None], valid=iivvviii, frames=['2', '4', '3']

readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(2)='2', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(1)='1', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(1)='1', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(7)='7', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

readPage(0)='0', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

page faults = 9, page ins = 9, page outs = 0

-------------- policy (write): OPT--------------

writePage(7, 'A'), frames=['A', None, None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(0, 'B'), frames=['A', 'B', None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(1, 'C'), frames=['A', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(2, 'D'), frames=['D', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(0, 'E'), frames=['D', 'E', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(3, 'F'), frames=['D', 'E', 'F'], swapSpace=['0', 'C', '2', '3', '4', '5', '6', 'A']

writePage(0, 'G'), frames=['D', 'G', 'F'], swapSpace=['0', 'C', '2', '3', '4', '5', '6', 'A']

writePage(4, 'H'), frames=['D', 'H', 'F'], swapSpace=['G', 'C', '2', '3', '4', '5', '6', 'A']

writePage(2, 'I'), frames=['I', 'H', 'F'], swapSpace=['G', 'C', '2', '3', '4', '5', '6', 'A']

writePage(3, 'J'), frames=['I', 'H', 'J'], swapSpace=['G', 'C', '2', '3', '4', '5', '6', 'A']

writePage(0, 'K'), frames=['I', 'K', 'J'], swapSpace=['G', 'C', '2', '3', 'H', '5', '6', 'A']

writePage(3, 'L'), frames=['I', 'K', 'L'], swapSpace=['G', 'C', '2', '3', 'H', '5', '6', 'A']

writePage(0, 'M'), frames=['I', 'M', 'L'], swapSpace=['G', 'C', '2', '3', 'H', '5', '6', 'A']

writePage(3, 'N'), frames=['I', 'M', 'N'], swapSpace=['G', 'C', '2', '3', 'H', '5', '6', 'A']

writePage(2, 'O'), frames=['O', 'M', 'N'], swapSpace=['G', 'C', '2', '3', 'H', '5', '6', 'A']

writePage(1, 'P'), frames=['O', 'M', 'P'], swapSpace=['G', 'C', '2', 'N', 'H', '5', '6', 'A']

writePage(2, 'Q'), frames=['Q', 'M', 'P'], swapSpace=['G', 'C', '2', 'N', 'H', '5', '6', 'A']

writePage(0, 'R'), frames=['Q', 'R', 'P'], swapSpace=['G', 'C', '2', 'N', 'H', '5', '6', 'A']

writePage(1, 'S'), frames=['Q', 'R', 'S'], swapSpace=['G', 'C', '2', 'N', 'H', '5', '6', 'A']

writePage(7, 'T'), frames=['T', 'R', 'S'], swapSpace=['G', 'C', 'Q', 'N', 'H', '5', '6', 'A']

writePage(0, 'U'), frames=['T', 'U', 'S'], swapSpace=['G', 'C', 'Q', 'N', 'H', '5', '6', 'A']

writePage(1, 'V'), frames=['T', 'U', 'V'], swapSpace=['G', 'C', 'Q', 'N', 'H', '5', '6', 'A']

page faults = 9, page ins = 9, page outs = 6

-------------- policy (read): LRU--------------

readPage(7)='7', pageTable=[None, None, None, None, None, None, None, 0], valid=iiiiiiiv, frames=['7', None, None]

readPage(0)='0', pageTable=[1, None, None, None, None, None, None, 0], valid=viiiiiiv, frames=['7', '0', None]

readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(4)='4', pageTable=[1, None, None, 2, 0, None, None, None], valid=viivviii, frames=['4', '0', '3']

readPage(2)='2', pageTable=[1, None, 2, None, 0, None, None, None], valid=viviviii, frames=['4', '0', '2']

readPage(3)='3', pageTable=[None, None, 2, 1, 0, None, None, None], valid=iivvviii, frames=['4', '3', '2']

readPage(0)='0', pageTable=[0, None, 2, 1, None, None, None, None], valid=vivviiii, frames=['0', '3', '2']

readPage(3)='3', pageTable=[0, None, 2, 1, None, None, None, None], valid=vivviiii, frames=['0', '3', '2']

readPage(0)='0', pageTable=[0, None, 2, 1, None, None, None, None], valid=vivviiii, frames=['0', '3', '2']

readPage(3)='3', pageTable=[0, None, 2, 1, None, None, None, None], valid=vivviiii, frames=['0', '3', '2']

readPage(2)='2', pageTable=[0, None, 2, 1, None, None, None, None], valid=vivviiii, frames=['0', '3', '2']

readPage(1)='1', pageTable=[None, 0, 2, 1, None, None, None, None], valid=ivvviiii, frames=['1', '3', '2']

readPage(2)='2', pageTable=[None, 0, 2, 1, None, None, None, None], valid=ivvviiii, frames=['1', '3', '2']

readPage(0)='0', pageTable=[1, 0, 2, None, None, None, None, None], valid=vvviiiii, frames=['1', '0', '2']

readPage(1)='1', pageTable=[1, 0, 2, None, None, None, None, None], valid=vvviiiii, frames=['1', '0', '2']

readPage(7)='7', pageTable=[1, 0, None, None, None, None, None, 2], valid=vviiiiiv, frames=['1', '0', '7']

readPage(0)='0', pageTable=[1, 0, None, None, None, None, None, 2], valid=vviiiiiv, frames=['1', '0', '7']

readPage(1)='1', pageTable=[1, 0, None, None, None, None, None, 2], valid=vviiiiiv, frames=['1', '0', '7']

page faults = 12, page ins = 12, page outs = 0

-------------- policy (write): LRU--------------

writePage(7, 'A'), frames=['A', None, None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(0, 'B'), frames=['A', 'B', None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(1, 'C'), frames=['A', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(2, 'D'), frames=['D', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(0, 'E'), frames=['D', 'E', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(3, 'F'), frames=['D', 'E', 'F'], swapSpace=['0', 'C', '2', '3', '4', '5', '6', 'A']

writePage(0, 'G'), frames=['D', 'G', 'F'], swapSpace=['0', 'C', '2', '3', '4', '5', '6', 'A']

writePage(4, 'H'), frames=['H', 'G', 'F'], swapSpace=['0', 'C', 'D', '3', '4', '5', '6', 'A']

writePage(2, 'I'), frames=['H', 'G', 'I'], swapSpace=['0', 'C', 'D', 'F', '4', '5', '6', 'A']

writePage(3, 'J'), frames=['H', 'J', 'I'], swapSpace=['G', 'C', 'D', 'F', '4', '5', '6', 'A']

writePage(0, 'K'), frames=['K', 'J', 'I'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(3, 'L'), frames=['K', 'L', 'I'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(0, 'M'), frames=['M', 'L', 'I'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(3, 'N'), frames=['M', 'N', 'I'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(2, 'O'), frames=['M', 'N', 'O'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(1, 'P'), frames=['P', 'N', 'O'], swapSpace=['M', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(2, 'Q'), frames=['P', 'N', 'Q'], swapSpace=['M', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(0, 'R'), frames=['P', 'R', 'Q'], swapSpace=['M', 'C', 'D', 'N', 'H', '5', '6', 'A']

writePage(1, 'S'), frames=['S', 'R', 'Q'], swapSpace=['M', 'C', 'D', 'N', 'H', '5', '6', 'A']

writePage(7, 'T'), frames=['S', 'R', 'T'], swapSpace=['M', 'C', 'Q', 'N', 'H', '5', '6', 'A']

writePage(0, 'U'), frames=['S', 'U', 'T'], swapSpace=['M', 'C', 'Q', 'N', 'H', '5', '6', 'A']

writePage(1, 'V'), frames=['V', 'U', 'T'], swapSpace=['M', 'C', 'Q', 'N', 'H', '5', '6', 'A']

page faults = 12, page ins = 12, page outs = 9

-------------- policy (read): FIFO--------------

readPage(7)='7', pageTable=[None, None, None, None, None, None, None, 0], valid=iiiiiiiv, frames=['7', None, None]

readPage(0)='0', pageTable=[1, None, None, None, None, None, None, 0], valid=viiiiiiv, frames=['7', '0', None]

readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(3)='3', pageTable=[None, 2, 0, 1, None, None, None, None], valid=ivvviiii, frames=['2', '3', '1']

readPage(0)='0', pageTable=[2, None, 0, 1, None, None, None, None], valid=vivviiii, frames=['2', '3', '0']

readPage(4)='4', pageTable=[2, None, None, 1, 0, None, None, None], valid=viivviii, frames=['4', '3', '0']

readPage(2)='2', pageTable=[2, None, 1, None, 0, None, None, None], valid=viviviii, frames=['4', '2', '0']

readPage(3)='3', pageTable=[None, None, 1, 2, 0, None, None, None], valid=iivvviii, frames=['4', '2', '3']

readPage(0)='0', pageTable=[0, None, 1, 2, None, None, None, None], valid=vivviiii, frames=['0', '2', '3']

readPage(3)='3', pageTable=[0, None, 1, 2, None, None, None, None], valid=vivviiii, frames=['0', '2', '3']

readPage(0)='0', pageTable=[0, None, 1, 2, None, None, None, None], valid=vivviiii, frames=['0', '2', '3']

readPage(3)='3', pageTable=[0, None, 1, 2, None, None, None, None], valid=vivviiii, frames=['0', '2', '3']

readPage(2)='2', pageTable=[0, None, 1, 2, None, None, None, None], valid=vivviiii, frames=['0', '2', '3']

readPage(1)='1', pageTable=[0, 1, None, 2, None, None, None, None], valid=vviviiii, frames=['0', '1', '3']

readPage(2)='2', pageTable=[0, 1, 2, None, None, None, None, None], valid=vvviiiii, frames=['0', '1', '2']

readPage(0)='0', pageTable=[0, 1, 2, None, None, None, None, None], valid=vvviiiii, frames=['0', '1', '2']

readPage(1)='1', pageTable=[0, 1, 2, None, None, None, None, None], valid=vvviiiii, frames=['0', '1', '2']

readPage(7)='7', pageTable=[None, 1, 2, None, None, None, None, 0], valid=ivviiiiv, frames=['7', '1', '2']

readPage(0)='0', pageTable=[1, None, 2, None, None, None, None, 0], valid=viviiiiv, frames=['7', '0', '2']

readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

page faults = 15, page ins = 15, page outs = 0

-------------- policy (write): FIFO--------------

writePage(7, 'A'), frames=['A', None, None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(0, 'B'), frames=['A', 'B', None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(1, 'C'), frames=['A', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(2, 'D'), frames=['D', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(0, 'E'), frames=['D', 'E', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(3, 'F'), frames=['D', 'F', 'C'], swapSpace=['E', '1', '2', '3', '4', '5', '6', 'A']

writePage(0, 'G'), frames=['D', 'F', 'G'], swapSpace=['E', 'C', '2', '3', '4', '5', '6', 'A']

writePage(4, 'H'), frames=['H', 'F', 'G'], swapSpace=['E', 'C', 'D', '3', '4', '5', '6', 'A']

writePage(2, 'I'), frames=['H', 'I', 'G'], swapSpace=['E', 'C', 'D', 'F', '4', '5', '6', 'A']

writePage(3, 'J'), frames=['H', 'I', 'J'], swapSpace=['G', 'C', 'D', 'F', '4', '5', '6', 'A']

writePage(0, 'K'), frames=['K', 'I', 'J'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(3, 'L'), frames=['K', 'I', 'L'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(0, 'M'), frames=['M', 'I', 'L'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(3, 'N'), frames=['M', 'I', 'N'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(2, 'O'), frames=['M', 'O', 'N'], swapSpace=['G', 'C', 'D', 'F', 'H', '5', '6', 'A']

writePage(1, 'P'), frames=['M', 'P', 'N'], swapSpace=['G', 'C', 'O', 'F', 'H', '5', '6', 'A']

writePage(2, 'Q'), frames=['M', 'P', 'Q'], swapSpace=['G', 'C', 'O', 'N', 'H', '5', '6', 'A']

writePage(0, 'R'), frames=['R', 'P', 'Q'], swapSpace=['G', 'C', 'O', 'N', 'H', '5', '6', 'A']

writePage(1, 'S'), frames=['R', 'S', 'Q'], swapSpace=['G', 'C', 'O', 'N', 'H', '5', '6', 'A']

writePage(7, 'T'), frames=['T', 'S', 'Q'], swapSpace=['R', 'C', 'O', 'N', 'H', '5', '6', 'A']

writePage(0, 'U'), frames=['T', 'U', 'Q'], swapSpace=['R', 'S', 'O', 'N', 'H', '5', '6', 'A']

writePage(1, 'V'), frames=['T', 'U', 'V'], swapSpace=['R', 'S', 'Q', 'N', 'H', '5', '6', 'A']

page faults = 15, page ins = 15, page outs = 12

-------------- policy (read): SecondChance--------------

readPage(7)='7', pageTable=[None, None, None, None, None, None, None, 0], valid=iiiiiiiv, frames=['7', None, None]

readPage(0)='0', pageTable=[1, None, None, None, None, None, None, 0], valid=viiiiiiv, frames=['7', '0', None]

readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0], valid=vviiiiiv, frames=['7', '0', '1']

readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None], valid=vvviiiii, frames=['2', '0', '1']

readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None], valid=vivviiii, frames=['2', '0', '3']

readPage(4)='4', pageTable=[1, None, None, 2, 0, None, None, None], valid=viivviii, frames=['4', '0', '3']

readPage(2)='2', pageTable=[None, None, 1, 2, 0, None, None, None], valid=iivvviii, frames=['4', '2', '3']

readPage(3)='3', pageTable=[None, None, 1, 2, 0, None, None, None], valid=iivvviii, frames=['4', '2', '3']

readPage(0)='0', pageTable=[2, None, 1, None, 0, None, None, None], valid=viviviii, frames=['4', '2', '0']

readPage(3)='3', pageTable=[2, None, 1, 0, None, None, None, None], valid=vivviiii, frames=['3', '2', '0']

readPage(0)='0', pageTable=[2, None, 1, 0, None, None, None, None], valid=vivviiii, frames=['3', '2', '0']

readPage(3)='3', pageTable=[2, None, 1, 0, None, None, None, None], valid=vivviiii, frames=['3', '2', '0']

readPage(2)='2', pageTable=[2, None, 1, 0, None, None, None, None], valid=vivviiii, frames=['3', '2', '0']

readPage(1)='1', pageTable=[2, 1, None, 0, None, None, None, None], valid=vviviiii, frames=['3', '1', '0']

readPage(2)='2', pageTable=[None, 1, 2, 0, None, None, None, None], valid=ivvviiii, frames=['3', '1', '2']

readPage(0)='0', pageTable=[0, 1, 2, None, None, None, None, None], valid=vvviiiii, frames=['0', '1', '2']

readPage(1)='1', pageTable=[0, 1, 2, None, None, None, None, None], valid=vvviiiii, frames=['0', '1', '2']

readPage(7)='7', pageTable=[0, None, 2, None, None, None, None, 1], valid=viviiiiv, frames=['0', '7', '2']

readPage(0)='0', pageTable=[0, None, 2, None, None, None, None, 1], valid=viviiiiv, frames=['0', '7', '2']

readPage(1)='1', pageTable=[0, 2, None, None, None, None, None, 1], valid=vviiiiiv, frames=['0', '7', '1']

page faults = 14, page ins = 14, page outs = 0

-------------- policy (write): SecondChance--------------

writePage(7, 'A'), frames=['A', None, None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(0, 'B'), frames=['A', 'B', None], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(1, 'C'), frames=['A', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', '7']

writePage(2, 'D'), frames=['D', 'B', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(0, 'E'), frames=['D', 'E', 'C'], swapSpace=['0', '1', '2', '3', '4', '5', '6', 'A']

writePage(3, 'F'), frames=['D', 'E', 'F'], swapSpace=['0', 'C', '2', '3', '4', '5', '6', 'A']

writePage(0, 'G'), frames=['D', 'G', 'F'], swapSpace=['0', 'C', '2', '3', '4', '5', '6', 'A']

writePage(4, 'H'), frames=['H', 'G', 'F'], swapSpace=['0', 'C', 'D', '3', '4', '5', '6', 'A']

writePage(2, 'I'), frames=['H', 'I', 'F'], swapSpace=['G', 'C', 'D', '3', '4', '5', '6', 'A']

writePage(3, 'J'), frames=['H', 'I', 'J'], swapSpace=['G', 'C', 'D', '3', '4', '5', '6', 'A']

writePage(0, 'K'), frames=['H', 'I', 'K'], swapSpace=['G', 'C', 'D', 'J', '4', '5', '6', 'A']

writePage(3, 'L'), frames=['L', 'I', 'K'], swapSpace=['G', 'C', 'D', 'J', 'H', '5', '6', 'A']

writePage(0, 'M'), frames=['L', 'I', 'M'], swapSpace=['G', 'C', 'D', 'J', 'H', '5', '6', 'A']

writePage(3, 'N'), frames=['N', 'I', 'M'], swapSpace=['G', 'C', 'D', 'J', 'H', '5', '6', 'A']

writePage(2, 'O'), frames=['N', 'O', 'M'], swapSpace=['G', 'C', 'D', 'J', 'H', '5', '6', 'A']

writePage(1, 'P'), frames=['N', 'P', 'M'], swapSpace=['G', 'C', 'O', 'J', 'H', '5', '6', 'A']

writePage(2, 'Q'), frames=['N', 'P', 'Q'], swapSpace=['M', 'C', 'O', 'J', 'H', '5', '6', 'A']

writePage(0, 'R'), frames=['R', 'P', 'Q'], swapSpace=['M', 'C', 'O', 'N', 'H', '5', '6', 'A']

writePage(1, 'S'), frames=['R', 'S', 'Q'], swapSpace=['M', 'C', 'O', 'N', 'H', '5', '6', 'A']

writePage(7, 'T'), frames=['R', 'T', 'Q'], swapSpace=['M', 'S', 'O', 'N', 'H', '5', '6', 'A']

writePage(0, 'U'), frames=['U', 'T', 'Q'], swapSpace=['M', 'S', 'O', 'N', 'H', '5', '6', 'A']

writePage(1, 'V'), frames=['U', 'T', 'V'], swapSpace=['M', 'S', 'Q', 'N', 'H', '5', '6', 'A']

page faults = 14, page ins = 14, page outs = 11