

Assignment 9

Due Date: Sunday, November 19, 2017, 11:59pm

Up to one-day late submission without penalty

Up to one-week late submission with 20% penalty

Submit electronically on iLMS

What to submit: One zip file named <studentID>-hw9.zip (replace <studentID> with your own student ID). It should contain four files:

- one PDF file named **hw9.pdf** for Section 1. Write your answers in English. Check your spelling and grammar. Include your name and student ID!
- Section 2: Python source files. Include your name and student ID in the program comments on top.
 - simplevm.py, typescript

1. [40 points] Problem Set

1. [20 points] **9.2** A simplified view of thread states is **Ready**, **Running**, and **Blocked**, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O). This is illustrated in Figure 9.30. Assuming a thread is in the Running state, answer the following questions, and explain your answer:
 - a. Will the thread change state if it incurs a page fault? If so, to what state will it change?
 - b. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what state will it change?
 - c. Will the thread change state if an address reference is resolved in the page table? If so, to what state will it change?

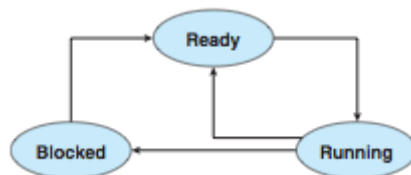


Fig. 9.30 Thread state diagram for Exercise 9.2

2. [10 points] **9.4** What is the copy-on-write feature, and under what circumstances is its use beneficial? What hardware support is required to implement this feature?
3. [10 points] **9.19** What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

2. [60 points] 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](#) 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' %
replacementPolichy)
        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 round robin
            # 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's an unknown policy
    raise ValueError('unknown policy %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, None, 0],
valid=iiiiiiiv, frames=['7', None, None]
readPage(0)='0', pageTable=[1, None, None, None, None, None, None, None, 0],
valid=viiiiiiiv, frames=['7', '0', None]
readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, None, 0],
valid=vviiiiiv, frames=['7', '0', '1']
readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None, None],
valid=vvviiii, frames=['2', '0', '1']
readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None, None],
valid=vvviiii, frames=['2', '0', '1']
readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None, None],
valid=vivviiii, frames=['2', '0', '3']
readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None, None],
valid=vivviiii, frames=['2', '0', '3']
readPage(4)='4', pageTable=[None, None, 0, 2, 1, None, None, None, None],
valid=iivvviii, frames=['2', '4', '3']

```

```

readPage(2)='2', pageTable=[None, None, 0, 2, 1, None, None, None],
valid=iiivviii, frames=['2', '4', '3']
readPage(3)='3', pageTable=[None, None, 0, 2, 1, None, None, None],
valid=iiivviii, frames=['2', '4', '3']
readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviii, frames=['2', '0', '3']
readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviii, frames=['2', '0', '3']
readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviii, frames=['2', '0', '3']
readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviii, frames=['2', '0', '3']
readPage(2)='2', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviii, frames=['2', '0', '3']
readPage(1)='1', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviii, frames=['2', '0', '1']
readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviii, frames=['2', '0', '1']
readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviii, frames=['2', '0', '1']
readPage(1)='1', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviii, frames=['2', '0', '1']
readPage(7)='7', pageTable=[1, 2, None, None, None, None, None, 0],
valid=vviiiiv, frames=['7', '0', '1']
readPage(0)='0', pageTable=[1, 2, None, None, None, None, None, 0],
valid=vviiiiv, frames=['7', '0', '1']
readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0],
valid=vviiiiv, 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=viiiiiiiv, 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=vvviiii, frames=['2', '0', '1']
readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviiii, 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=vvviiii, frames=['1', '0', '2']
readPage(1)='1', pageTable=[1, 0, 2, None, None, None, None, None],
valid=vvviiii, 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=viiiiiiiv, 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=vvviiii, frames=['2', '0', '1']
readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviiii, 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=viivviiii, frames=['4', '3', '0']
readPage(2)='2', pageTable=[2, None, 1, None, 0, None, None, None],
valid=viviviiii, frames=['4', '2', '0']
readPage(3)='3', pageTable=[None, None, 1, 2, 0, None, None, None],
valid=iivvviiii, 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=vvviviiii, frames=['0', '1', '3']

```



```

readPage(2)='2', pageTable=[0, 1, 2, None, None, None, None, None],
valid=vvviii, frames=['0', '1', '2']
readPage(0)='0', pageTable=[0, 1, 2, None, None, None, None, None],
valid=vvviii, frames=['0', '1', '2']
readPage(1)='1', pageTable=[0, 1, 2, None, None, None, None, None],
valid=vvviii, frames=['0', '1', '2']
readPage(7)='7', pageTable=[None, 1, 2, None, None, None, None, 0],
valid=ivviii, frames=['7', '1', '2']
readPage(0)='0', pageTable=[1, None, 2, None, None, None, None, 0],
valid=viviii, frames=['7', '0', '2']
readPage(1)='1', pageTable=[1, 2, None, None, None, None, None, 0],
valid=vviii, 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=vviiiiv, frames=['7', '0', '1']
readPage(2)='2', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviiiiv, frames=['2', '0', '1']
readPage(0)='0', pageTable=[1, 2, 0, None, None, None, None, None],
valid=vvviiiiv, frames=['2', '0', '1']
readPage(3)='3', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviiiiv, frames=['2', '0', '3']
readPage(0)='0', pageTable=[1, None, 0, 2, None, None, None, None],
valid=vivviiiiv, frames=['2', '0', '3']
readPage(4)='4', pageTable=[1, None, None, 2, 0, None, None, None],
valid=viiivviiiiv, frames=['4', '0', '3']
readPage(2)='2', pageTable=[None, None, 1, 2, 0, None, None, None],
valid=iivvviiiiv, frames=['4', '2', '3']
readPage(3)='3', pageTable=[None, None, 1, 2, 0, None, None, None],
valid=iivvviiiiv, frames=['4', '2', '3']
readPage(0)='0', pageTable=[2, None, 1, None, 0, None, None, None],
valid=vivviviiiiv, frames=['4', '2', '0']
readPage(3)='3', pageTable=[2, None, 1, 0, None, None, None, None],
valid=vivviviiiiv, frames=['3', '2', '0']
readPage(0)='0', pageTable=[2, None, 1, 0, None, None, None, None],
valid=vivviviiiiv, frames=['3', '2', '0']
readPage(3)='3', pageTable=[2, None, 1, 0, None, None, None, None],
valid=vivviviiiiv, frames=['3', '2', '0']
readPage(2)='2', pageTable=[2, None, 1, 0, None, None, None, None],
valid=vivviviiiiv, frames=['3', '2', '0']
readPage(1)='1', pageTable=[2, 1, None, 0, None, None, None, None],
valid=vvviviiiiv, frames=['3', '1', '0']
readPage(2)='2', pageTable=[None, 1, 2, 0, None, None, None, None],
valid=ivvviiiiv, frames=['3', '1', '2']
readPage(0)='0', pageTable=[0, 1, 2, None, None, None, None, None],
valid=vvviiiiv, frames=['0', '1', '2']
readPage(1)='1', pageTable=[0, 1, 2, None, None, None, None, None],
valid=vvviiiiv, 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=vviiiiv, 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

```