Name: LessTanker

Collaborators: Name1, Name2

Problem 1-1.

(a) $(f_5, f_3, f_4, f_2, f_1)$

(b) $(\{f_1, f_2\}, f_5, f_4, f_3)$

(c) $(\{f_2, f_5\}, f_4, f_3, f_1)$

(d) $(f_5, f_3, f_4, f_2, f_1)$

Problem 1-2.

(a) In a for loop from j=0 to k-1, call insert_at(i+k-j-1,delete_at(i)).
My idea is in a loop, each time we move one element from front to the end, then the first place's index will always be i, and the last index to reverse is correspond to j, and the index to insert in is i+k-j-1.

(b) In a for loop from it=0 to k-1,call insert_at(j-it-1,delete_at(i+k-it-1)).
Similiar to the above one, my solution is to delete the original list from the end, then move the element to the front of index j.

Problem 1-3. My solution is a linked list with index inside the value.

For initialization, obviously a linked list takes O(x) time to do. place_mark(i,m) can even go to O(1) time, while read_page(i),shift_mark(m,d) and move_page(m) is just basic linked list operation which all takes O(1) time.

Problem 1-4.

(a) insert_first(x) needs to let head.prev = x , x.next = head, x.prev = null.
insert_last(x) needs to let tail.next = x , x.prev = tail, x.next=null.
delete_first() needs to set head.next.prev = null.
delete_last() needs to set last.prev.next = null.

(b) let a = x1.next, b=x2.prev.
Then set a.prev = null, b.next=null.
Next,let x1.next = x2, x2.prev = x1
Finally, return a as the head of the new doubly linked list, b as the tail.

(c) Remember x.next as y,then set x.next = L2.head, L2.head.prev = x,L2.tail.next = y, y.prev=L2.tail.

(d) Submit your implementation to alg.mit.edu.