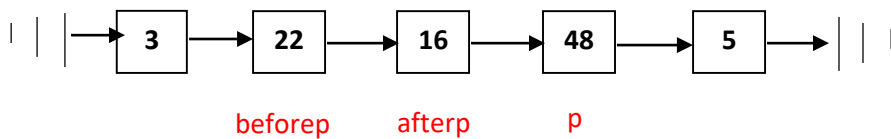
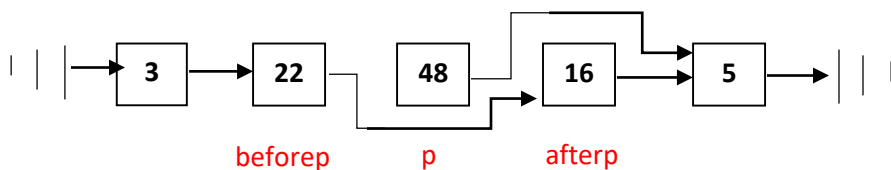
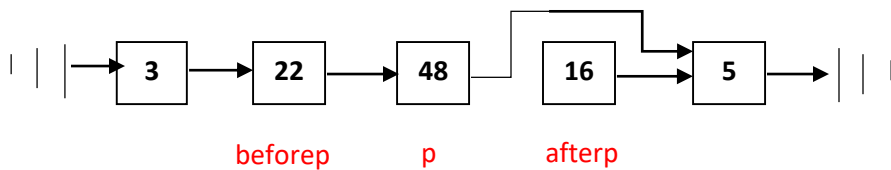
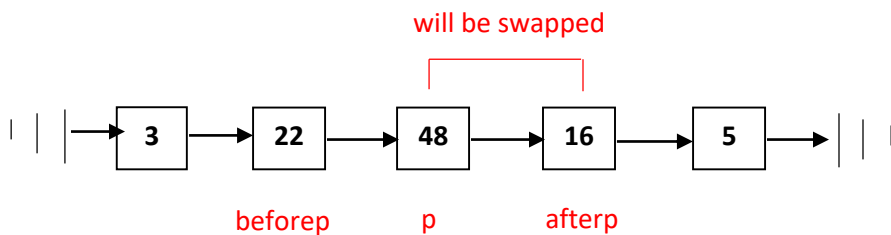


Recitation 2 - Analysis & Linked Lists

3.2 Swap two adjacent elements by adjusting only the links (and not the data)
using: a) Singly linked lists, b) Doubly linked lists

a) `// beforep is before the two adjacent cells that are to be
// swapped. beforep has to be given
// Error checks are omitted for clarity
void swapWithNext(Node beforep) {
 Node p, afterp;
 p = beforep.next;
 afterp = p.next; // both p and afterp assumed not NULL
 p.next = afterp.next;
 beforep.next = afterp;
 afterp.next = p; }`



b) // p and afterp are to be switched. Giving p is enough

```
void swapWithNext(Node p) {
    Node beforep, afterp;
    beforep = p.prev;
    afterp = p.next;
    p.next = afterp.next;
    beforep.next = afterp;
    afterp.next = p;
    p.next.prev = p;
    p.prev = afterp;
    afterp.prev = beforep; }
```

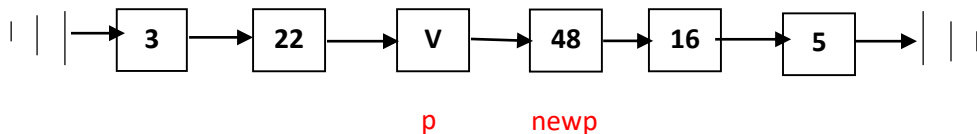
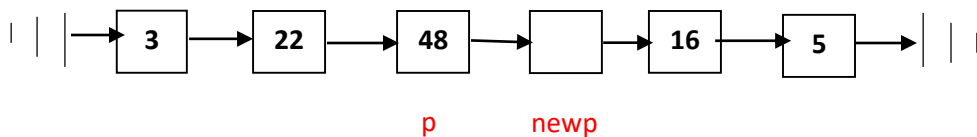
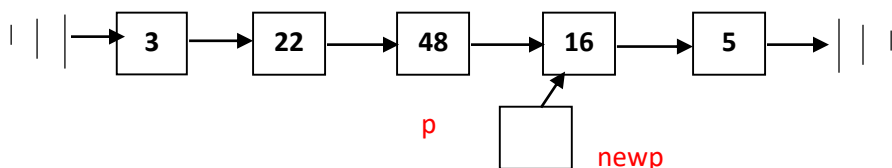
3.37 Suppose that a singly linked list is implemented with both a header and tail node. Describe constant-time algorithms to: (15 minutes at max.)

a) Insert item x before position p (given by an iterator)

b) Remove the item stored at position p (given by an iterator)

a) void insertBefore(Node p, int newValue) {

```
    Node newp = new Node();
    newp.next = p.next;
    p.next = newp;
    newp.data = p.data;
    p.data = newValue; }
```



```
b) void deleteNode (Node p) {  
    p.data = p.next.data;  
    p.next = p.next.next; }  
}
```

