3.3//suppose that we are swapping the xth and (x+1)th element;

a

```
struct node{
	int element;
	node *next;
};
void swap()
{
	node *p,*head,*temp;
	p=head;
	for(int i=0;i<x-1;i++)
	p=p->next;
	temp=p->next->next;
	temp->next=p->next;
	p->next->next=p->next->next->next;
	p->next=temp;
}
```

b

```
struct node{
	int element;
	node *next;
	node *prev;
};
void swap()
{
	node *p,*head,*temp;
	p=head;
	for(int i=0;i<x+1;i++)
	p=p->next;
	temp=p->prev;
	temp->prev->next=p;
	p->prev=temp->prev;
	temp->next=p->next;
	temp->prev=p;
	p->next->prev=temp;
	p->next=temp;
}
```

3.4

```
void intersection(node *head1,*head2)
{
	node *p,*q,*temp;
	p=head1->next;
```

```
    q=head2->next;
    node *head;
    temp=head;
    while(p!=NULL&&q!=NULL)
    {
        if(p->element<q->element)
        p=p->next;
        else if(p->element>q->element)
        q=q->next
        else
        {
            p->next=temp->next;
            temp->next=p;
            p=p->next;
            q=q->next;
        }
    }
}
```

3.15

a

```
void adjust(int a[1000],int find)
{
    int position;
    for(int i=0;;i++)
    {
        if(a[i]==find)
        {
            position=i;
            break;
        }
    }
    for(int i=position-1;i>=0;i--)
    a[i+1]=a[i];
    a[0]=find;
}
```

b

```
void adjust(node *head,int find)
{
    node *p,*q;
    p=head;
    while(1)
    {
```

```
            if(p->next->element==find)
            break;
            p=p->next;
        }
        q=(node *)malloc(sizeof(node));
        q=p->next;
        q=head->next;
        head->next=q;
        p->next=p->next->next;
}
```

3.17

Advantage: It is easier to understand and practise;

Disadvantage: It requires more time and memory;

```
void lazy_delete(node *p,find)
{
        int positive=0,negative=0;
        if(DELETE)
        {
            positive++;
            flag=1;
        }
        else
        negative++;
        if(positive==negative)
        {
            while(head->next!=NULL)
            {
                if(head->next->flag)
                DELETE(head);
                else
                head=head->next;
            }
        }
}
```