# Ordered Linked Lists

```
NODEPTR insert(NODEPTR list, int x)
{
        NODEPTR prev, curr, q ;
        if (list == NULL){
                list =insrthead (list, x) ;
                return list;
        }
        prev = curr = list ;
        while((curr!=NULL) && (curr->info<=x))
           if (curr->info==x){
                printf(" Data found \n") ;
                exit(1);
           }
           else {
                prev=curr; curr= curr->next ;
              }
        }
        if (prev==curr)       list = insrthead (list, x);
        else                  insafter (prev, x);
        return (list);
}
```

- NODEPTR merge ( NODEPTR p, NODEPTR q )

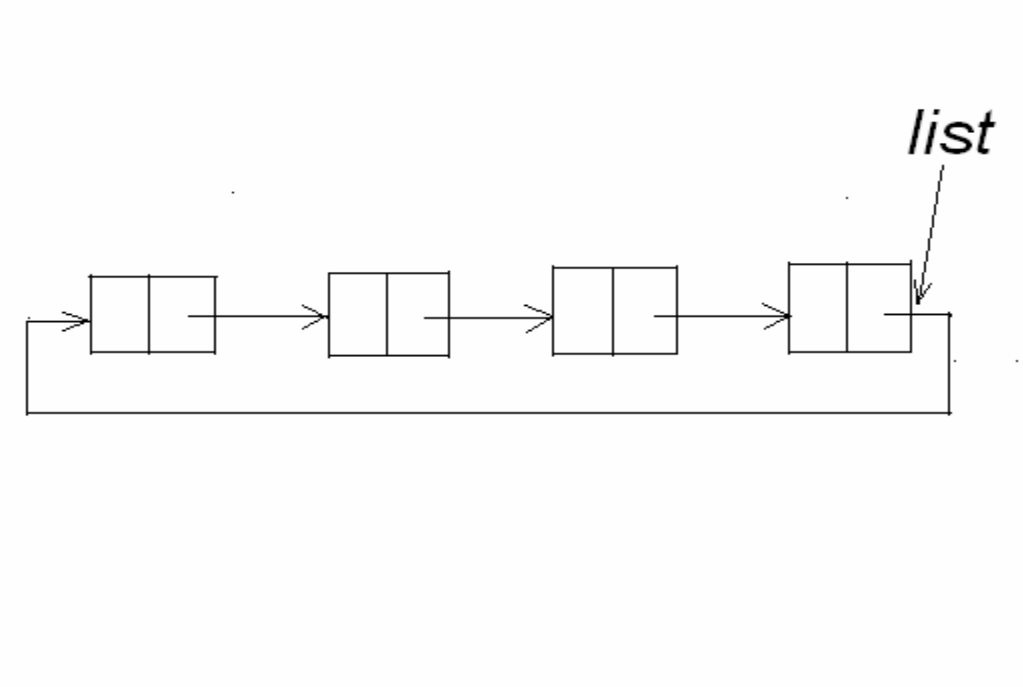
```
{
  NODEPTR p1,q1,r,r1,r2 ;
  p1=p; q1=q;
  if ((p1==NULL)&&(q1==NULL))
  {   printf("Null lists to be merged") ;
      exit(1);
  };
 r = r1 = NULL;
```

```
while ((p1!=NULL)&&(q1!=NULL){
        r2 = getnode( );
        if (r == NULL)  r =r2;
        if (p1->info<=q1->info){
                r2->info = p1->info;
                r2->next = NULL ;
                p1= p1->next;
         }
         else
         {      r2->info = q1->info;
                r2->next = NULL ;
                q1 = q1->next;
         }
         if (r1!=NULL) r1->next = r2
         r1= r2 ;
}/* end while */
```

```
while(p1!= NULL){
    r2 = getnode();
    if (r == NULL) r=r2;
    r2->info = p1->info ;
    r2->next = NULL;
    p1 = p1->next;
    if(r1!=NULL){
            r1->next = r2;
    }
    r1= r2;
}
```

```
while(q1!= NULL){
        r2 = getnode();
        if (r == NULL) r=r2;
        r2->info = q1->info ;
        r2->next = NULL;
        q1 = q1->next;
        if(r1!=NULL){
                r1->next = r2;
        }
        r1= r2;
}
```

# Circular Lists



# Concatenating Two Lists

```
void concat (NODEPTR *plist1, NODEPTR *plist2)
{       NODEPTR p;
        if(*plist2==NULL)
                retrun ;
        if(*plist1==NULL){
                *plist1=*plist2;
                return;
        }
        p=(*plist1)->next;
        (*plist1)->next=(*plist2)->next;
        (*plist2)->next=p;
        *plist1=*plis2;
        return;
}/*end concat   concat(&list1,&list2)*/
```

- **Doubly Linked Lists**
- struct node{

```
        int info;
        struct node *left,*right ;
  };
typedef struct node *NODEPTR ;
```

```
void delete (NODEPTR p, int *px)
{       NODEPTR q, r;
        if (p == NULL){
                printf("Void deletion \n");
                return ;
        }/*end if*/
        *px=p->info;
        q=p->left;
        r=p->right;
        q->right=r;
        r->left=q;
        freenode(p);
        return;
}/*end delete*/
```

```
void insertright (NODPETR p, int x)
{       NODEPTR q, r;
        if (p == NULL){
                printf ("Void insertion \n");
                return;
        }/*end if */
        q =getnode();
        q->info=x;
        r=p->right;
        r->left=q;
        q->right=r;
        q->left=p;
        p->right=q;
        return;
}/* end inserion*/
```