```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE
NODE getnode()
{
  NODE x;
x = (NODE)malloc(sizeof(struct node));
if (x==NULL)
  {
    pf ("mem full");
      exit(0);
  }
  return x;
}
void freenode(NODE x)
  {
     free (x);
  }
NODE insert_front(NODE first, int item)
{
   NODE temp;
   temp=getnode();
   temp->info = item;
   temp->link = NULL;
  if (first ==NULL)
    return temp;
  temp->link = first;
   first = temp;
    return first;
}
```

```
delete_front (NODE first)
{
   NODE temp;
   if (first==NULL)
     {
       pf(" list is empty");
       return first;
     }
   temp = first;
   temp = temp->link;
pf("the deleted item is %d", temp->i
 free(first);
 return temp;
}
NODE insert_rear(NODE first, int item)
{
   NODE temp, cur;
  temp = getnode();
  temp->info = item;
  temp->link = NULL;
  if (first==NULL)
     return temp;
    cur = first;
  while (cur->link!=NULL)
  cur = cur->link;
   cur->link = temp;
     return first;
}
```

```
NODE delete_rear (NODE first)
{
    NODE prev, cur;
    if (first == NULL)
    {
        Pf ("List empty");
        return first;
    }
    if (first -> link == NULL)
    {
        Pf ("Item deleted %d", first->info
        free (first);
        return NULL;
    }
    prev = NULL;
    cur = first;
    while (cur -> link != NULL)
    {
        prev = cur;
        cur = cur -> link;
    }
    Pf ("Item deleted at rear end %d", cur->in
    free (cur);
    prev -> link = NULL;
    return first;
}
```

```
NODE concat(NODE first, NODE second)
{
    NODE cur;
    if (first == NULL)
        return second;
    if (second == NULL)
        return first;
    cur = first;
    while (cur->link != NULL)
        cur = cur->link;
    cur->link = second;
    return first;
}
```

```
NODE reverse(NODE first)
{
    NODE cur, temp;
    cur = NULL
    while (first != NULL)
    {
        temp = first;
        first = first->link;
        temp->link = cur;
        cur = temp;
    }
    return cur;
}
```

```
ordered list_(int item, NODE first)
{
  NODE temp, prev, cur;
   temp = getnode();
   temp -> info = item;
  temp -> link = NULL;
if (first == NULL)
    return temp;
  if (item < first -> info)
 {
 temp -> link = first;
   return temp;
 }
  prev = NULL;
  cur = first;
while (cur != NULL && item > cur -> info)
  { prev = cur;
   cur = cur -> link;
   }
  prev -> link = temp;
   temp -> link = cur;
   return first;
}
```