```
#include <stdio.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)
  {
    printf("Creation Unsuccessfull");
    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;
}

Node delete_front (NODE first)
{
node temp;
if (first == NULL)
{
printf ("List is empty \n");
return first;
}.

temp = first;
temp = temp -> link;
printf (" item deleted at front
 and is %d ", first -> info);
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 cur, prev;
  if (first == NULL)
  printf ("list is empty; cannot delete \n");
  return first;
}
  if (first -> link == NULL)
  {
    printf ("item deleted is %d \n", first -> info);
  free (first);
  return NULL;
}
prev = NULL
cur = first;
```

```
while (cur->link != NULL)
{
prev = cur;
cur = cur->link;
}
printf("The item deleted at rear end is
    %d", cur->info);
free(cur);
prev->link = NULL;
return first;
}
Node insert_pos (int item, int pos, NODE
first)
{
Node temp;
Node prev, cur;
int count;
temp = getnode();
temp->info = item;
temp->link = NULL;
if (first == NULL && pos == 1)
return temp;
if (first == NULL)
{
printf("Invalid");
return first;
}
```

```
if (pos == 2)
{
  temp->link = first;
  return temp;
}

count = 1;
prev = NULL;
cur = first;
while (cur != NULL && count != pos)
{
  prev = cur;
  cur = cur->link;
  count++;
}

if (count == pos)
 {
  prev->link = temp;
  temp->link = cur;
  return first;
}

printf ("IP\n");
return first;

}
```

```
Node delete_pos (int pos, NODE first)
 if (first == NULL) {
 printf ("list is empty \n");
 return first;
 }
 Node temp = first;
 if (pos == 1)
 {
  first = temp -> link;
  free (temp);
 return first;
}

NODE prev;

for(int i = 1, temp != NULL && i < pos; i++)
  {
    prev = temp;
    temp = temp -> link;
  }

if (temp == NULL || temp -> link == NULL)
  {
    printf("Invalid position\n");
     return NULL;
  }

prev -> link = temp -> link;
printf("Element deleted %d\n", temp -> info);
free(temp);
```

```
return first;
}

void display(NODE first)
{
  NODE temp;
  if (first == NULL)
  printf("list empty cannot display items\n");
  for(temp = first; temp != NULL; temp = temp->link)
  {
   printf("%d\n", temp -> info);
  }
}

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;
}
```

(5)

```
Node reverse (Node first)
{
  Node cur, temp;
  cur = NULL;
  while (first != NULL)
  {
    temp = first;
    first = first -> link;
    temp -> link = cur;
    cur = temp;
  }
  return cur;
}
void main()
{
NODE sort (Node first)
Node current = first;
Node index = NULL;
int temp;
if (head == NULL){
  return;
}
else {
while (current != NULL)
```

```
{
    index = current -> link;

while (index != NULL) {

if (current -> data > index -> data)
    {
      temp = current -> data;
      current -> data = index -> data;
      index -> data = temp;
    }

index = index -> link;
}

current = current -> link;
}
}
return first current;
}

void main()
{
int item, choice, pos, i, n;
Node a, b;
Node first = NULL;

for(;;)
{
printf("1. insert_front \n2. delete_front \n3. insert_
      rear \n4. delete_rear \n5. insert at pos \n6.
```

```
delete at pos \n 7. concat \n 8. reverse \n
9. sort \n 10. display \n");

printf("enter the choice \n");
scanf("%d", &choice);
switch(choice)
{

case 1: printf("enter the item at
front end \n");
scanf("%d", &item);

first = insert_front(first, item);
break;

case 2: first = delete_front(first);
break;

case 3: printf("Enter the item at rear-
end \n");
scanf("%d", &item);
first = insert_rear(first, item);
break;

case 4: first = delete_rear(first);
break;

case 5:
printf("Enter item \n");
scanf("%d", &item);
```

```
printf("Enter position \n");
scanf("%d", &pos);
first = insert_pos(item, pos, first);
break;

case 6:
printf("Enter position of deletion \n");
scanf("%d", &pos);
first = delete_pos(pos, first);
break;

case 7:

printf("Enter the no. of nodes in 1 \n");
    scanf("%d", &n);
  a = NULL;
for(i=0; i<n; i++)
  {
    printf("Enter the item \n");
    scanf("%d", &item);
    a = insert_rear(a, item);
  }
printf("Enter the no. of nodes in 2 \n");
  scanf("%d", &n);
b = NULL;
for(i=0; i<n; i++)
  {
    printf("Enter the item \n");
    scanf("%d", &item);
    b = insert_rear(b, item);
  }
```

```
a = concat(a, b);
display(a);
break;

case 8:

first = remove(first);
    display(first);
    break;

case 9: sort(first)

first = sort(first)
display(first);
break;


case 10: display(first);
        break;

default: exit(0);
  break;
}
}
}
```