IBM19CS021

URBAN EDGE

```
#include <stdlib.h>
#include <string.h>
Struct node
{
   int sem;
   Struct node *next;
};
struct node *head = NULL;
Struct node *head2 = NULL;
int c=0;
void Insert()
{
 Struct node * newnode;
 Struct node * temp;
 int s;
 printf("Enter Integer : ");
 Scanf("%d",&s);
 newnode = (struct node*) malloc (sizeof( struct
 newnode →sem = s;
 if (head == NULL)
 {
 newnode → next = NULL;
  head = newnode;
  printf("first node is created");
  c++;
 }
 else
 {
 temp = head;
 While (temp →next != NULL)
  {
    temp = temp → next;
  }
```

```
temp->next = newnode;
newnode->next = NULL;
ct++;
printf("Node created");
}
}
void Insert2()
{
struct node *newnode;
struct node *temp;
int s,y;
printf("Enter elements to create list 2\n");
do
{
printf("Enter integer:\n");
scanf("%d",&s);
newnode = (struct node*)malloc(sizeof(struct node));
newnode->sem=s;
if(head2==NULL)
{
newnode->next=NULL;
head2=newnode;
printf("first node of LL");
ct++;
}
else{
temp=head2;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
newnode->next=NULL;
ct++;
```

```
        printf("\n Node created");
    {
    printf("\n continue 0 or 1 ");
    scanf("%d",&y);
    } while(y!=0);
}

void bubbleSort()
{
 int swapped, i;
 struct node *ptr1;
 struct node *lptr = NULL;
 if (head == NULL)
   return;

 do
 {
   swapped = 0;
   ptr1 = head;
   while (ptr1->next != lptr)
   {
     if (ptr1->sem > ptr1->next->sem)
     {
       int temp = ptr1->sem;
       ptr1->sem = ptr1->next->sem;
       ptr1->next->sem = temp;
       swapped = 1;
     }
     ptr1 = ptr1->next;
   }
   lptr = ptr1;

 } while(swapped)
}
```

```
void reverse()
{
 struct node * prev = NULL;
 struct node * current = head;
 struct node * next = NULL;
 while(current != NULL)
 {
  next = current -> next;
  current -> next = prev;
  prev = current;
  current = next;
 }
  head = prev;
}

void concat()
{
 struct node * ptr;
  if (head == NULL)
   {
     head = head2;
  }
  if (head2 == NULL)
   {
     head2 = head;
    }
  ptr = head;
   while (ptr -> next != NULL)
         ptr = ptr -> next;
    ptr -> next = head2;
}
```

```
void display1()
{
 struct node *ptr;
 ptr = head;
 if (ptr == NULL)
 {
 printf("\n List is Empty");
 }
 else
 {
 while (ptr != NULL)
 {
 printf("%d", ptr->sem);
 ptr= ptr->next;
 }
}

int main()
{
 int choice, pos;
 do
 {
 printf("1. Insert node
 2. Sort node
 3. reverse
 4. Concat.
 5. exit"
 )
 scanf("%d", &choice);
 switch (choice)
 {
 case 1:
 Insert()
 break;
```

```
case 2:
    printf ("before : \n");
    display1();
    bubblesort();
    print ("after \n");
    display1();
    break;
case 3:
    printf ("before : \n")
    display1();
    reverse();
    printf ("after : \n");
    display1();
case 4:
    Insert2();
    concat();
    display1();
    break;
case 5:
    break;
}
} while (choice != 5);
return 0;
}
```