```c
//Merging 2 sorted lists represented using
//circular singly linked lists

typedef struct node *Nodeptr;
struct node{
    int data;
    Nodeptr next;
};

int IsEmpty(Nodeptr last){
    return (last == NULL ? 1:0);
}

Nodeptr getnode(){
    Nodeptr temp;
    temp = (Nodeptr) malloc(sizeof(struct node));
    return temp;
}
void Display(Nodeptr last){
    Nodeptr temp;

    if (IsEmpty(last)){ printf("Empty List"); return;  }
    printf("Contents of Circular List : \n");
    temp=last->next;
    printf("%d\n",temp->data);
    while(temp != last){
        temp = temp->next;
        printf("%d\n",temp->data);
    }
}
```

```
void InsertLast(Nodeptr *last, int x){
     Nodeptr temp;

     temp= getnode();
     temp->data = x;

     if (IsEmpty(*last)){
          temp->next = temp;
          (*last) = temp;
     }
     else{
          temp->next = (*last)->next;
          (*last)->next = temp;
          (*last) = temp;
     }
}
```

```
Nodeptr Create(){
    Nodeptr last=NULL;
    int x;

    printf("Enter elements of the list (in asc order) [-1 to End]");
    scanf("%d",&x);
    while(x!=-1){
        InsertLast(&last,x);
        scanf("%d",&x);
    }
    return last;
}
```

```
int Length(Nodeptr last){
    int n=0;
    Nodeptr temp = last;
    if (IsEmpty(last)) return 0;
    do{
        n++;
        temp = temp->next;
    }while(temp!=last);
    return n;
}
```

```
Nodeptr merge(Nodeptr lastA, Nodeptr lastB){
    Nodeptr lastC = NULL;
    int m,n;

    if (IsEmpty(lastA))
        return lastB;
    if (IsEmpty(lastB))
        return lastA;
    m = Length(lastA);
    n = Length(lastB);
    Nodeptr first1 = lastA->next;
    Nodeptr first2 = lastB->next;
```

```
int i=0,j=0;
while(i<m && j<n){//while there are nodes
    if (first1->data<first2->data){
        InsertLast(&lastC,first1->data);
        first1 = first1->next;
        i++;
    }
    else{
        InsertLast(&lastC,first2->data);
        first2 = first2->next;
        j++;
    }
}

   while (i<m){ //while there are nodes remaining in first list
       InsertLast(&lastC,first1->data);
       first1 = first1->next;
       i++;
   }

   while (j<n){//while there are nodes remaining in second list
       InsertLast(&lastC,first2->data);
       first2 = first2->next;
       j++;
   }

   return lastC;
}
```

```c
int main(){
    Nodeptr a=NULL,b=NULL,c=NULL;

    a = Create();
    b = Create();

    Display(a);
    Display(b);

    c = merge(a,b);
    Display(c);

    return 0;
}
```