```c
#include <stdio.h>
#include <stdlib.h>

typedef int ListElementType;

typedef struct ListNode *ListPointer;
typedef struct ListNode
{
    ListElementType Data;
    ListPointer Next;
} ListNode;

typedef enum {
    FALSE, TRUE
} boolean;


void CreateList(ListPointer *List);
boolean EmptyList(ListPointer List);
void LinkedInsert(ListPointer *List, ListElementType Item, ListPointer PredPtr);
void LinkedDelete(ListPointer *List, ListPointer PredPtr);
void LinkedTraverse(ListPointer List);
void LinearSearch(ListPointer List, ListElementType Item, ListPointer *PredPtr, boolean *Found);
void OrderedLimearSearch(ListPointer List, ListElementType Item, ListPointer *PredPtr, boolean *Found);
void Inverse_List(ListPointer *List);

int main()
{
    ListPointer AList,PredPtr;
    ListElementType Item;
    int i,n;

    CreateList(&AList);

    printf("Give the number of integers:");
    scanf("%d",&n);

    for(i = 0; i < n; i++)
    {
        printf("Give an integer:");
        scanf("%d",&Item);
        PredPtr = NULL;
        LinkedInsert(&AList,Item,PredPtr);
    }

    LinkedTraverse(AList);

    Inverse_List(&AList);

    LinkedTraverse(AList);

    return 0;
}

void CreateList(ListPointer *List)
{
    *List = NULL;
}

boolean EmptyList(ListPointer List)
{
    return (List==NULL);
}

void LinkedInsert(ListPointer *List, ListElementType Item, ListPointer PredPtr)
{
```

```
    ListPointer TempPtr;

   TempPtr= (ListPointer)malloc(sizeof(struct ListNode));
 /*  printf("Insert &List %p, List %p,  &(*List) %p, (*List) %p, TempPtr %p\n",
   &List, List,  &(*List), (*List), TempPtr); */
   TempPtr->Data = Item;
    if (PredPtr==NULL) {
        TempPtr->Next = *List;
        *List = TempPtr;
    }
    else {
        TempPtr->Next = PredPtr->Next;
        PredPtr->Next = TempPtr;
    }
}

void LinkedDelete(ListPointer *List, ListPointer PredPtr)
{
    ListPointer TempPtr;

    if (EmptyList(*List))
        printf("EMPTY LIST\n");
   else
   {
        if (PredPtr == NULL)
        {
              TempPtr = *List;
              *List = TempPtr->Next;
        }
        else
        {
             TempPtr = PredPtr->Next;
             PredPtr->Next = TempPtr->Next;
        }
        free(TempPtr);
    }
}

void LinkedTraverse(ListPointer List)
{
    ListPointer CurrPtr;

   if (EmptyList(List))
        printf("EMPTY LIST\n");
   else
   {
        CurrPtr = List;
         while ( CurrPtr!=NULL )
        {
             printf("%d ",(*CurrPtr).Data);
             CurrPtr = CurrPtr->Next;
        }
        printf("\n");
   }
}

void LinearSearch(ListPointer List, ListElementType Item, ListPointer *PredPtr, boolean *Found)
{
   ListPointer CurrPtr;
   boolean stop;

   CurrPtr = List;
    *PredPtr=NULL;
   stop= FALSE;
   while (!stop && CurrPtr!=NULL )
    {
```

```
          if (CurrPtr->Data==Item )
             stop = TRUE;
          else
          {
             *PredPtr = CurrPtr;
             CurrPtr = CurrPtr->Next;
          }
     }
     *Found=stop;
}

void OrderedLimearSearch(ListPointer List, ListElementType Item, ListPointer *PredPtr, boolean *Found)
{
   ListPointer CurrPtr;
   boolean DoneSearching;

   CurrPtr = List;
   *PredPtr = NULL;
   DoneSearching = FALSE;
   *Found = FALSE;
   while (!DoneSearching && CurrPtr!=NULL )
    {
          if (CurrPtr->Data>=Item )
          {
             DoneSearching = TRUE;
             *Found = (CurrPtr->Data==Item);
          }
          else
          {
             *PredPtr = CurrPtr;
             CurrPtr = CurrPtr->Next;
          }
     }
}

void Inverse_List(ListPointer *List)
{
    ListPointer HelpList,CurrPtr;

    /*????????? ????? ??? ?? ????????? ?? ???????? ??? ??? ??????.*/
    CreateList(&HelpList);

    if(!EmptyList(*List))
    {
        /*??????????? ??? ?????? ????? ??? ??? ???? ??? ??? ?????????? ???????
        ??? ??? ?? ???????? ???? ????????? ?????.*/
        CurrPtr=*List;
        while(CurrPtr->Next != NULL)
        {
            LinkedInsert(&HelpList,CurrPtr->Data,NULL);
            CurrPtr=CurrPtr->Next;
        }
        /*??????? ??? ?? ????????? ????????.*/
        LinkedInsert(&HelpList,CurrPtr->Data,NULL);
    }
    /*??????????? ??? ????????? ???? ?????? ?????.*/
    *List=HelpList;
}
```