```
#include    <stdio.h>                                                    //"ArrayListmain.cpp"
#include    <stdlib.h>
#include    "ArrayList.h"
#include    "List.h"
#include    "ArrayList.cpp"

int WhoIsPred(Ldata d1, Ldata d2){
    if(d1<d2) return 0;
    return 1;
}

int main(void) {
    List*    MyList= (List *)malloc(sizeof(List)) ; int  a, b ;
    Ldata   Sdata, Bdata, A[10]= {6,2,7,9,8,3,5,4,0,1} ;

    InitList(MyList) ;                  //리스트의 초기화
    LInsert(MyList,5) ;   LInsert(MyList,3) ;   LInsert(MyList,1) ;//리스트에 자료 추가

    printf("%3d %3d %3d \n",        //리스트 자료의 출력
        MyList->Array[0] , MyList->Array[1] , MyList->Array[2]) ;

    SetSortRule(MyList, WhoIsPred) ;
//-------------------------------  //정렬 기준 적용------------
    b= MyList->NumOfData ;          //기존 자료 정렬
    MyList->NumOfData= 0;
    for(a=0; a<b; a++){
        LInsert(MyList, MyList->Array[a]) ;
    }
//-------------------------------------------------------------
    for (a=0; a<10; a++) {
        LInsert(MyList, A[a]) ;         //리스트에 자료 추가
    }
    LPrint(MyList) ;                    //리스트 자료의 출력

    b= MyList->NumOfData-1 ;
    if(LFirst(MyList, &Sdata)) {
        Bdata= Sdata ;
        while(MyList->CurPosition < b && LNext(MyList, &Sdata)) {
            if(Bdata == Sdata) {
                printf("중복 자료 %d 제거 \n", LRemove(MyList)) ;
                b-- ;
            }
            Bdata= Sdata ;
        }
    }
    LPrint(MyList) ;                    //리스트 자료의 출력
}
```

```
#include    <stdio.h>                                                    //"ArrayList.cpp"
#include    <string.h>
#include    "ArrayList.h"
#include    "List.h"

void    InitList(List *plist) {                         //리스트의 초기화
    memset(plist->Array, 0, sizeof(Ldata)*LEN_List) ;
    plist->NumOfData= 0 ;
    plist->comp= NULL;
}

void    SInsert(List *plist, Ldata pdata) {             //정렬 삽입
    int Cur= LCount(plist) ;
    while(Cur>0 && plist->comp(plist->Array[Cur-1], pdata)) {   //앞의 자료가 기준 자료보다 작을 때까지
        plist->Array[Cur]=plist->Array[Cur-1] ;         //앞의 자료를 뒤 자리로 옮기고
        Cur-- ;                                         //자리를 찾을 때까지 반복
    }
    plist->Array[Cur]= pdata ;                          //찾아진 자리에 자료 저장
}
```

```
void    LInsert(List *plist, Ldata pdata) {            //자료 삽입
    int Cur= LCount(plist) ;
    if(Cur >= LEN_List) {
        printf("리스트가 꽉차서 자료를 추가할 수 없습니다\n") ;
        return ;
    }
    if(Cur && plist->comp)  SInsert(plist, pdata) ;
    else plist->Array[Cur]= pdata ;

    plist->NumOfData++ ;
}

void    LPrint(List *plist) {
    int a= 0 ;

    while(a<plist->NumOfData) {
        printf("%3d ", plist->Array[a++]) ;
    } printf("\n") ;
}
```

```
int  LFirst(List *plist, Ldata *pdata) {    //첫 자료?
    if(LCount(plist)) {                     //있음
        plist->CurPosition= 0 ;
        *pdata= plist->Array[0] ;    //첫 자료 값
        return 1 ;
    }
    return 0 ;
}
```

```
int  LNext(List *plist, Ldata *pdata) {  //다음 자료?
    if(plist->CurPosition < LCount(plist)) {   //있음
        plist->CurPosition++;
        *pdata= plist->Array[plist->CurPosition] ;
        return 1 ;
    }
    return 0 ;
}
```

```
Ldata LRemove(List *plist) {                    //자료 삭제
    if(LCount(plist)){
        int rpos= plist->CurPosition ;          //삭제할 위치
        Ldata rdata= plist->Array[rpos] ;       //삭제할 자료
        plist->CurPosition= rpos-1 ;            //삭제한 후의 CurrentPosition

        while(rpos < plist->NumOfData) {    //자료의 끝까지 지워진 자리 채움
            plist->Array[rpos]= plist->Array[rpos+1] ;
            rpos++ ;
        }
        plist->NumOfData-- ;
        return rdata ;                      //삭제된 자료 반환
    }
}

int  LCount(List *plist){
    return plist->NumOfData ;
}

void SetSortRule(List *plist, int (*comp)(Ldata, Ldata)){
    plist->comp= comp;
}
```

```
#ifndef  _LIST_H_                           //"List.h"
#define _LIST_H_

void    InitList(List *) ;          //리스트의 초기화
void    Linsert(List *, Ldata) ;    //자료 삽입
int     LFirst(List *, Ldata *) ;   //첫 자료?
int     LNext(List *, Ldata *) ;    //다음 자료?
Ldata   LRemove(List *) ;           //자료 삭제
void    LPrint(List *) ;            //리스트 자료 출력
int     LCount(List *) ;            //리스트 자료 개수
void    SetSortRule(List    *,    int    (*comp)(Ldata,
Ldata)) ;         //정렬 규칙 설정

#endif
```

```
#ifndef  _ARRAY_LIST_H_             //"ArrayList.h"
#define _ARRAY_LIST_H_

#define LEN_List    50

typedef     int  Ldata ;

typedef struct _arraylist {
    Ldata   Array[LEN_List] ;   //자료 저장 배열
    int     CurPosition ;       //현재 인덱스
    int     NumOfData ;         //저장 자료 개수
    int     (*comp)(Ldata, Ldata) ;
} ArrayList ;

typedef ArrayList   List ;
#endif
```