```
#include    <stdio.h>                                                    //"PointListmain.cpp"
#include    "Point.h"
#include    "List.h"
#include    "PointList.cpp"

int WhoIsPred(Ldata d1, Ldata d2){
    if(d1->x < d2->x)       return 0;
    else if(d1->x == d2->x)
        if(d1->y < d2->y)   return 0;
        else                return 1;
    else                    return 1;
}

Ldata NewPoint(int x, int y) {
    Ldata   Npoint= (Point *)malloc(sizeof(Point)) ;
    Npoint->x= x ;  Npoint->y= y ;  Npoint->Next= NULL ;
    return Npoint;
}

void Sorting(List *plist) {
    SetSortRule(plist, WhoIsPred) ;
    int     a, su= 0 ;
    Ldata   Tmp, Addr[100] ;

    Tmp= plist->Head ;
    while(Tmp= Tmp->Next) {       //첫 번째 자료부터 모든 자료의 포인터를
        Addr[su++]= Tmp ;             //배열에 저장
    }
    plist->Head->Next->Next= NULL ;   //첫 자료의 Next를 NULL로 설정
    plist->NumOfData= 1 ;

    for(a=1; a<su; a++) {             //두 번째 자료 이후부터 모든 자료를
        Addr[a]->Next= 0 ;            //Next 포인터를 NULL로 설정하여
        LInsert(plist, Addr[a]) ;     //리스트에 추가(Sorting Rule 적용)
    }
}

int main(void) {
    List*   MyList= (List *)malloc(sizeof(List)) ;
    int     a, A[10]= {6,2,7,9,8,3,5,4,0,1} ;
    Ldata   point= (Point *)malloc(sizeof(Point)), rpoint ;
    InitList(MyList) ;                    //리스트의 초기화
    printf("\n새로운 노드를 리스트의 앞에 추가 \n") ;
    LInsert(MyList, NewPoint(1,2)) ; LInsert(MyList, NewPoint(2,3)) ; LInsert(MyList, NewPoint(3,4)) ;
    LPrint(MyList) ;
//  SetSortRule(MyList, WhoIsPred) ;
    printf("\n오름차순 Sorting Rule 적용 자료 추가 \n") ;
    Sorting(MyList) ;
    for (a=0; a<10; a+=2)   {         //리스트에 자료 추가
        LInsert(MyList, NewPoint(A[a],A[a+1])) ;
        LPrint(MyList) ;
    }
    printf("\n좌표에 3인 포함된 모든 점 삭제 \n") ;
    if(LFirst(MyList, point)) {
        if(point->x == 3 || point->y == 3){
            rpoint= LRemove(MyList) ;
            printf("(%d,%d) 제거 \n", rpoint->x, rpoint->y) ;     free(rpoint) ;
        }
        while(LNext(MyList,point)){
            if(point->x == 3 || point->y == 3){
                rpoint= LRemove(MyList) ;
                printf("(%d,%d) 제거 \n", rpoint->x, rpoint->y) ; free(rpoint) ;
            }
        }
    }
    printf("\n남아 있는 모든 점 리스트 \n") ;
    LPrint(MyList) ;                      //리스트 자료의 출력
}
```

```
#include     <stdio.h>          //"PointListmain.cpp"
#include     <stdlib.h>
#include     "Point.h"
#include     "List.h"

void     InitList(List *plist) {      //리스트의 초기화
     Ldata Head= (Point *)malloc(sizeof(Point)) ;
     Head->Next= NULL ;
     plist->Head= Head ;
     plist->Head->Next= NULL ;
     plist->NumOfData= 0 ;
     plist->comp= NULL;
}
```

```
void     PrintNode(Ldata pnode) {
     printf("(%d,%d) ", pnode->x, pnode->y);
}

void     LPrint(List *plist) {
     if(LCount(plist)) {
          Ldata   pdata= plist->Head->Next ;

          while(pdata) {
               PrintNode(pdata) ;
               pdata= pdata->Next ;
          } printf("\n") ;
     }
}
```

```
void     SInsert(List *plist, Ldata pdata) {
     Ldata Before= plist->Head ;

     while(Before->Next && plist->comp(pdata,Before->Next)) {
          Before= Before->Next ;
     }
     pdata->Next= Before->Next ;
     Before->Next= pdata ;
}
```

```
void     LInsert(List *plist, Ldata pdata) {
     if(plist->comp)
          SInsert(plist, pdata) ;
     else {
          pdata->Next= plist->Head->Next ;
          plist->Head->Next= pdata ;
     }
     plist->NumOfData++ ;
}
```

```
int  LFirst(List *plist, Ldata pdata) {      //첫 자료?
     if(LCount(plist)) {
          plist->Before= plist->Head ;
          plist->Cur= plist->Head->Next ;
          pdata->x= plist->Cur->x ;
          pdata->y= plist->Cur->y ;
          return 1 ;  //첫 자료 있음
     }
     return 0 ;       //저장된 자료가 없음
}
```

```
int  LNext(List *plist, Ldata pdata) {//다음 자료?
     if(plist->Cur->Next) {
          plist->Before= plist->Cur ;
          plist->Cur= plist->Cur->Next ;
          pdata->x= plist->Cur->x ;
          pdata->y= plist->Cur->y ;
          return 1 ;  //다음 자료 있음
     }
     return 0 ;       //다음 자료 없음
}
```

```
Ldata LRemove(List *plist) {                    //자료 삭제
     if(LCount(plist)) {
          Ldata rpos= plist->Cur ;              //삭제할 위치

          plist->Before->Next= rpos->Next ;
          plist->Cur= plist->Before ;           //삭제한 후의 CurrentPosition
          plist->NumOfData-- ;                  //자료 개수 감소
          return rpos ;                         //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
                                     //정렬 규칙
void     SetSortRule(List *, int (*comp)(Ldata, Ldata)) ;

#endif
```

```
#ifndef  _POINT_H_          //"Point.h"
#define _POINT_H_

typedef struct _point {
     int x, y ;
     struct _point *Next ;
} Point ;
typedef Point    *Ldata ;

typedef struct _list {
     Ldata Head, Before, Cur ;
     int  NumOfData ;
     int  (*comp)(Ldata, Ldata) ;
} List ;

#endif
```