```c
/* A sample solution of make-up quiz */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAME_LENGTH 20        /* maximum length of name */

typedef struct {
    char name[MAX_NAME_LENGTH];
    int age;
} child_t;                     /* a struct type for representing children in a kindergarten */

typedef struct linklist_st{
    child_t data;
    struct linklist_st * next;
} linklist_t;                   /* a struct type for link list */

/* Adds n children records to a binary file */
void addNewRecordsToFile(char * filename, int n);

/* Prints the records in the file */
void listRecords(char * filename);

/* Constructs a link-list by using the records in the file */
void recordsToList(char * filename, linklist_t ** list);

/* Print the information of children in the link-list */
void printList(linklist_t * list);

/* Do not forget to free the memory that you take for the list */
void freeList(linklist_t * list);

/* Sorts the records in the list according to names and grades. If two children have the same names, the younger one will come first in the sorted list*/
void sortList(linklist_t ** list);

int main()
{
    linklist_t * list = NULL;

    //addNewRecordsToFile("children.dat", 10);
    printf("\n##########################################################\n");

    listRecords("children.dat");

    recordsToList("children.dat", &list);
    printf("\n##########################################################\n");
    printList(list);

    printf("\n!#########################################################\n");
    sortList(&list);
    printf("\n##########################################################\n");
    printList(list);

    freeList(list);

    return 0;
}

/* Adds n children records to a binary file */
void addNewRecordsToFile(char * filename, int n)
{
    int i;
    child_t child;
    FILE * file = fopen(filename, "ab+");

    if (!file)            /* check if the file is opened or not */
    {
        printf("File %s could not be opened\n", filename);
        exit(-1);
    }

    for (i=0;i<n;++i)
    {
        printf("Name of child for record - %2d/%2d: ", i+1, n);
        scanf("%s", child.name);
        printf("Age of child for record  - %2d/%2d: ", i+1, n);
        scanf("%d", &child.age);

        fwrite(&child, sizeof(child_t), 1, file);
    }

    fclose(file);    /* do not forget to close your file */

}

/* Prints the records in the file */
void listRecords(char * filename)
{
    child_t child;
    FILE * file = fopen(filename, "rb");

    if (!file)
    {
        printf("File %s could not be opened\n", filename);
        exit(-1);
    }

    while (fread(&child, sizeof(child_t), 1, file)) {
        printf("%20s%5d\n", child.name, child.age);
    }

    fclose(file);
}

/* Sorts the records in the list according to names and grades. If two children have 
  the same names, the younger one will come first in the sorted list*/
void sortList(linklist_t ** list)  {
    linklist_t * prev = NULL;
    linklist_t * cur = *list;
    linklist_t * next = cur->next;
    int kont = 1;

    while (kont) {
        kont = 0;
        prev = NULL;
        cur = *list;
        next = cur->next;
        while (next)
        {
            if (strcmp(cur->data.name,next->data.name)>0)
            {
                kont = 1;
                if (!prev)
                {
                    *list = next;
                }
                else {
                    prev->next = next;
                }
                cur->next = next->next;
                next->next = cur;
            }
            else if (!strcmp(cur->data.name,next->data.name) && (cur->data.age > 
            next->data.age))
            {
                kont = 1;
                if (!prev)
                {
                    *list = next;
                }
                else {
                    prev->next = next;
                }
                cur->next = next->next;
                next->next = cur;
            }

            prev = cur;
            cur = next;
            next = cur->next;
        }
    }
}

/* Constructs a link-list by using the records in the file */
void recordsToList(char * filename, linklist_t ** list)
{
    child_t child;
    linklist_t * curPos, * nextPos;
    FILE * file = fopen(filename, "rb");

    if (!file)
    {
        printf("File %s could not be opened\n", filename);
        exit(-1);
    }


    while (fread(&child, sizeof(child_t), 1, file)) {
        curPos = (linklist_t*)malloc(sizeof(linklist_t));
        curPos->data.age = child.age;
        strcpy(curPos->data.name, child.name);
        curPos->next = NULL;

        if (!*list)
        {
            *list = curPos;
            nextPos = curPos;
        }
        else {
            nextPos->next = curPos;
            nextPos = curPos;
        }
    }

    fclose(file);
}

/* Print the information of children in the link-list */
void printList(linklist_t * list)
{
    if (!list)
        return;

    printf("%20s%5d\n", list->data.name, list->data.age);

    printList(list->next);
}

/* Do not forget to free the memory that you take for the list */
void freeList(linklist_t * list)
{
   if (!list)
        return;

    freeList(list->next);

    free(list);
}
```