BIL 108 - Computer Programming
Makeup Labwork

25.05.2012

In this week we will study on linked lists and binary files. We provide you with a partly implemented code in the appendix. You will use this code and implement some missing functions in the code.

PART 1(2 Pts) Implement the function **`recordsToList`** which will read the binary file created by the given code into a linked list.

PART 2 (2Pts) Implement the function **`sortList`** which will sort the linked list you have created in part 1 with respect to the names and ages in priority order(i.e., consider ages if and only if the names of the records are the same).

Appendix:

```
#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);

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

        freeList(list);

        return 0;
}
```

```c
/* 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)  {

}

/* 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)
{
        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);
}
```