

```
1  /* A sample solution of make-up quiz */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  #define MAX_NAME_LENGTH 20          /* maximum length of name */
7
8  typedef struct {
9      char name[MAX_NAME_LENGTH];
10     int age;
11 } child_t;                          /* a struct type for representing children in a
    kindergarten */
12
13 typedef struct linklist_st{
14     child_t data;
15     struct linklist_st * next;
16 } linklist_t;                       /* a struct type for link list */
17
18 /* Adds n children records to a binary file */
19 void addNewRecordsToFile(char * filename, int n);
20
21 /* Prints the records in the file */
22 void listRecords(char * filename);
23
24 /* Constructs a link-list by using the records in the file */
25 void recordsToList(char * filename, linklist_t ** list);
26
27 /* Print the information of children in the link-list */
28 void printList(linklist_t * list);
29
30 /* Do not forget to free the memory that you take for the list */
31 void freeList(linklist_t * list);
32
33 /* 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*/
34 void sortList(linklist_t ** list);
35
36 int main()
37 {
38     linklist_t * list = NULL;
39
40     //addNewRecordsToFile("children.dat", 10);
41     printf("\n#####\n");
42
43     listRecords("children.dat");
44
45     recordsToList("children.dat", &list);
46     printf("\n#####\n");
47     printList(list);
48
49     printf("\n!#####\n");
50     sortList(&list);
51     printf("\n#####\n");
52     printList(list);
53
54     freeList(list);
```

```
55
56     return 0;
57 }
58
59 /* Adds n children records to a binary file */
60 void addNewRecordsToFile(char * filename, int n)
61 {
62     int i;
63     child_t child;
64     FILE * file = fopen(filename, "ab+");
65
66     if (!file)                /* check if the file is opened or not */
67     {
68         printf("File %s could not be opened\n", filename);
69         exit(-1);
70     }
71
72     for (i=0;i<n;++i)
73     {
74         printf("Name of child for record - %2d/%2d: ", i+1, n);
75         scanf("%s", child.name);
76         printf("Age of child for record - %2d/%2d: ", i+1, n);
77         scanf("%d", &child.age);
78
79         fwrite(&child, sizeof(child_t), 1, file);
80     }
81
82     fclose(file);    /* do not forget to close your file */
83
84 }
85
86 /* Prints the records in the file */
87 void listRecords(char * filename)
88 {
89     child_t child;
90     FILE * file = fopen(filename, "rb");
91
92     if (!file)
93     {
94         printf("File %s could not be opened\n", filename);
95         exit(-1);
96     }
97
98     while (fread(&child, sizeof(child_t), 1, file)) {
99         printf("%20s%5d\n", child.name, child.age);
100     }
101
102     fclose(file);
103 }
104
105 /* Sorts the records in the list according to names and grades. If two children have the
    the same names, the younger one will come first in the sorted list*/
106 void sortList(linklist_t ** list) {
107     linklist_t * prev = NULL;
108     linklist_t * cur = *list;
109     linklist_t * next = cur->next;
```

```
110     int kont = 1;
111
112     while (kont) {
113         kont = 0;
114         prev = NULL;
115         cur = *list;
116         next = cur->next;
117         while (next)
118         {
119             if (strcmp(cur->data.name,next->data.name)>0)
120             {
121                 kont = 1;
122                 if (!prev)
123                 {
124                     *list = next;
125                 }
126                 else {
127                     prev->next = next;
128                 }
129                 cur->next = next->next;
130                 next->next = cur;
131             }
132             else if (!strcmp(cur->data.name,next->data.name) && (cur->data.age >
133                 next->data.age))
134             {
135                 kont = 1;
136                 if (!prev)
137                 {
138                     *list = next;
139                 }
140                 else {
141                     prev->next = next;
142                 }
143                 cur->next = next->next;
144                 next->next = cur;
145             }
146             prev = cur;
147             cur = next;
148             next = cur->next;
149         }
150     }
151 }
152
153 /* Constructs a link-list by using the records in the file */
154 void recordsToList(char * filename, linklist_t ** list)
155 {
156     child_t child;
157     linklist_t * curPos, * nextPos;
158     FILE * file = fopen(filename, "rb");
159
160     if (!file)
161     {
162         printf("File %s could not be opened\n", filename);
163         exit(-1);
164     }
```

```
165
166
167     while (fread(&child, sizeof(child_t), 1, file)) {
168         curPos = (linklist_t*)malloc(sizeof(linklist_t));
169         curPos->data.age = child.age;
170         strcpy(curPos->data.name, child.name);
171         curPos->next = NULL;
172
173         if (!*list)
174         {
175             *list = curPos;
176             nextPos = curPos;
177         }
178         else {
179             nextPos->next = curPos;
180             nextPos = curPos;
181         }
182     }
183
184     fclose(file);
185 }
186
187 /* Print the information of children in the link-list */
188 void printList(linklist_t * list)
189 {
190     if (!list)
191         return;
192
193     printf("%20s%5d\n", list->data.name, list->data.age);
194
195     printList(list->next);
196 }
197
198 /* Do not forget to free the memory that you take for the list */
199 void freeList(linklist_t * list)
200 {
201     if (!list)
202         return;
203
204     freeList(list->next);
205
206     free(list);
207 }
```